Computational Biology: Genomes, Networks, Evolution
MIT course 6.047/6.878

Taught by Prof. Manolis Kellis

September 11, 2013

CONTENTS

Preface and Acknowledgements

These notes summarize the material taught in the MIT course titled "Computational Biology: Genomes, Networks, Evolution", also cross-listed with Harvard, HST, HSPH and BU over the years. The course was listed as MIT course 6.047/6.878 in 2007-2011 (and under the temporary numbers 6.085/6.095/6.895 in Fall 2005-2006, and 6.096 in Spring 2005). It was cross-listed with MIT/Harvard Health Sciences and Technology (HST) course HST.507 in 2007-2011, Boston University Biological Engineering course BE-562 in 2008 and 2009, and Harvard School of Public Health course IMI231 in 2009-2011.

The course was originally developed by Prof. Manolis Kellis at MIT, with advice from **Silvio Micali**. It was first taught in Spring Spring 2005 as a half-course extension to the Introduction to Algorithms Course (6.046), and as an independent full-credit course in Fall 2005-2011. The course was co-lectured with Prof. **Piotr Indyk** in Fall 2005-2006, who contributed to the material on hashing and dimentionality reduction techniques. It was co-taugh with Prof. **James Galagan** in Fall 2007-2009 who contributed to the lectures on expression clustering, supervised learning and metabolic modelling, and who continued teaching the course independently at BU.

The material in the course has benefited tremendously from courses by **Bonnie Berger** at MIT, whose course "Introduction to Computational Biology (18.417)" was co-taught by Manolis Kellis as a student in Fall 2001, and **Serafim Batzoglou** at Stanford whose course "Computational Genomics (CS262)" was an inspiration for clarity and style and a source of figures and diagrams for the early chapters on alignment and HMMs. Lastly, the material in the course also benefited from two books used extensively in the course in the last several years, titled "Biological Sequence Analysis" by **Durbin, Eddy, Drogh, and Mitchison**, and "Bioinformatics Algorithms" by **Jones and Pevzner**.

The material of several chapters was initially developed by guest lecturers who are experts in their field and contributed new material, figures, slides, organization, and thoughts in the form of one or more lectures. Without them, the corresponding chapters would not have been possible. They are: **Pardis Sabeti** (Population Genetic Variation), **Mark Daly** (Medical Genetics), **David Reich** (Population History), **Eric Alm** (Bacterial Genomics), **John Rinn** (Long Non-Coding RNAs), **James Galagan** (Steady State modeling), **Matt Rasmussen** (Phylogenomics), **Mike Lin** (Gene finding), **Stefan Washietl** (RNA folding), **Jason Ernst** (Epigenomics), **Sushmita Roy** (Regulatory Networks), **Pouya Kheradpour** (Regulatory Genomics).

The Teaching Assistants who taught recitations and help develop the course problem sets have been **Reina Reimann** (Spring 2005), **Pouya Kheradpour** (Fall 2005), **Matt Rasmussen** and **Mike Lin** (Fall 2006), **Mike Lin** and **David Sontag** (Fall 2007), **Matt Rasmussen** and **Pouya Kheradpour** (Fall 2008), **Ed Reznik** and **Bob Altshuler** (Fall 2009), **Matt Edwards** (Fall 2010), and **Melissa Gymrek** (Fall 2011). The notes were originally compiled in a uniform format **Anna Shcherbina** (Fall 2011).

The current and past members of the **MIT CompBio Lab** (http://compbio.mit.edu/people.html), who have taught me as they grew into experts in their own fields. They are: Matt Rasmussen, Mike Lin, Pouya Kheradpour, Alexander Stark, Xiaohui Xie, Jason Ernst, Sushmita Roy, Luke Ward, Chris Bristow, Abdoulaye Diallo, David Hendrix, Loyal Goff, Stefan Washietl, Daniel Marbach, Mukul Bansal, Matthew Eaton, Irwin Jungreis, Rachel Sealfon, Bob Altshuler, Jessica Wu, Angela Yen, Soheil Feizi, Luis Barrera, Ben Holmes, Anna Ayuso, Wouter Meuleman, Ferhat Ay, Rogerio Candeias, Patrick Meyer, Tom Morgan, Wes Brown, Will Gibson, Rushil Goel, Luisa Di Stefano, Stephan Ossowski, Aviva Presser, Erez Lieberman, Joshua Grochow, Yuliya Kodysh, Leopold Parts, Ameya Deoras, Matt Edwards, Adrian Dalca.

The students taking the class and contributing to the scribe notes are:

- Spring 2005: Dan Arlow, Arhab Battacharyya, Punyashloka Biswal, Adam Bouhenguel, Dexter Chan, Shuvo Chatterjee, Tiffany Dohzen, Lyric Doshy, Robert Figueiredo, Edena Gallagher, Josh Grochow, Aleksas Hauser, Blanca Himes, George Huo, Xiaoming Jia, Scott Johnson, Steven Kannan, Faye Kasemset, Jason Kelly, Daniel Kim, Yuliya Kodysh, Nate Kushman, Lucy Mendel, Jose Pacheco, Sejal Patel, Haiharan Rahul, Gireeja Ranade, Sophie Rapoport, Aditya Rastogi, Shubhangi Saraf, Oded Shaham, Walter Stiehl, Kevin Stolt, James Sun, Xin Sun, Kah Tai, Kah Tay, Chester Tse, Verlik Tzanov, Brian Wu

- Fall 2005: Ebad Ahmed, Christophe Falling, Michael Farry, Elaine Gee, Luke Hutchison, Michael Lin, Grigore Pintilie, Asfandyar Qureshi, Matthew Rasmussen, Alexandru Salcianu, Zeeshan Syed, Hayden

Taylor, Velin Tzanov, Grant Wang

- Fall 2006: Mats Ahlgren, Zhu Ailing, Bob Altshuler, Nada Amin, Shay Artzi, Solomon Bisker, Allen Bryan, Sumeet Gupta, Adam Kiezun, Richard Koche, Mieszko Lis, Ryan Newton, Michael O'Kelly, Chris Reeder, Jonathan Rhodes, Michael Schnall-Levin, Alex Tsankov, Tarak Upadhyaya, Kush Varshney, Sam Volchenboum, Jon Wetzel, Amy Williams

- Fall 2007: Anton Aboukhalil, Matthew Belmonte, Ellenor Brown, Brad Cater, Alal Eran, Guilherme Fujiwara, Saba Gul, Kate Hoff, Shannon Iyo, Eric Jonas, Peter Kruskall, Michael Lee, Ben Levick, Fulu Li, Alvin Liang, Joshua Lim, Chit-Kwan Lin, Po-Ru Loh, Kevin Modzelewski, Georgis Papachristoudis, Michalis Potamias, Emmanuel Santos, Alex Schwendner, Maryam Shanechi, Timo Somervuo, James Sun, Xin Sun, Robert Toscano, Qingqing Wang, Ning Xie, Qu Zhang, Blaine Ziegler

- Fall 2008: Burak Alver, Tural Badirkhanli, Arnab Bhattacharyya, Can Cenik, Clara Chan, Lydia Chilton, Arkajit Dey, Ardavan Farjadpour, Jeremy Fineman, Bernhard Haeupler, Arman Hajati, Ethan Heilman, Joe Herman, Irwin Jungreis, Arjun Manrai, Nilah Monnier, Christopher Rohde, Rachel Sealfon, Daniel Southern, Paul Steiner, David Stiebel, Mengdi Wang

- Fall 2009: Layla Barkal, Michael Bennie, David Charlton, Guoliang Chew, John Dong, Matthew Edwards, Eric Eisner, Subha Gollakota, Nathan Haseley, Allen Lin, Christopher McFarland, Michael Melgar, Anrae Motes, Anand Oza, Elizabeth Perley, Brianna Petrone, Arya Tafvizi Zavareh, Yi-Chieh Wu, Angela Yen, Morteza Zadimoghaddam, Chelsea Zhang, James Zou

- Fall 2010: Minjeong Ahn, Andreea Bodnari, Wesley Brown, Jenny Cheng, Bianca Dumitrascu, Sam Esfahani, Amer Fejzic, Talitha Forcier, Maria Frendberg, Dhruv Garg, Rushil Goel, Melissa Gymrek, Benjamin Holmes, Wui Ip, Isaac Joseph, Geza Kovacs, Gleb Kuznetsov, Adam Marblestone, Alexander Mccauley, Sheida Nabavi, Jacob Shapiro, Andrew Shum, Ashutosh Singhal, Mark Smith, Mashaal Sohail, Eli Stickgold, Tahin Syed, Lance Wall, Albert Wang, Fulton Wang, Jerry Wang

- Fall 2011: Asa Adadey, Leah Alpert, Ahmed Bakkar, Rebecca Bianco, Brett Boval, Kelly Brock, Peter Carr, Efrain Cermeno, Alex Chernyakhovsky, Diana Chien, Akashnil Dutta, Temuge Enkhbaatar, Maha Farhat, Alec Garza-Galindo, Fred Grober, Gabriel Ha, Marc Hafner, Neel Hajare, Timothy Helbig, Ivan Imaz, Yarden Katz, Gwang Ko, David Ku, Yu-Chi Kuo, Dan Landay, Yinqing Li, Mark Mimee, Selene Mota, Hyun Ji Noh, Chrisantha Perera, Aleksey Pesterev, Michael Quintin, Maria Rodriguez, Megan Roytman, Abhishek Sarkar, Angela Schwarz, Meriem Sefta, Anna Shcherbina, Mindy Shi, Noam Shoresh, Eric Soderstrom, Ying Qi Soh, Sarah Spencer, Derrick Sund, Ruqi Tang, Zenna Tavares, Arvind Thiagarajan, Paul Tillberg, Christos Tzamos, Leonardo Urbina, Manasi Vartak, Nathan Villagaray-Carski, Sajith Wickramasekara, Thomas Willems, Maxim Wolf, Lok Sang Wong, Iris Xu, Johannes Yeh, Deniz Yorukoglu, Boyang Zhao.

CHAPTER

ONE

INTRODUCTION TO THE COURSE

TODO: missing *@scribe: who was this scribed by?*

Figures

1.1 Introduction and Goals

1.1.1 A course on computational biology

These lecture notes are aimed to be taught as a term course on computational biology, each 1.5 hour lecture covering one chapter, coupled with bi-weekly homework assignments and mentoring sessions to help students accomplish their own independent research projects. The notes grew out of MIT course 6.047/6.878, and very closely reflect the structure of the corresponding lectures.

1.1.2 Duality of Goals: Foundations and Frontiers

There are two goals for this course. The first goal is to introduce you to the **foundations** of the field of computational biology. Namely, introduce the fundamental biological problems of the field, and learn the algorithmic and machine learning techniques needed for tackling them. This goes beyond just learning how to use the programs and online tools that are popular any given year. Instead, the aim is for you to understand

the underlying principles of the most successful techniques that are currently in use, and provide you with the capacity to design and implement the next generation of tools. That is the reason why an introductory algorithms class is set as a pre-req; the best way to gain a deeper understanding for the algorithms presented is to implement them yourself.

The second goal of the course is to tackle the research **frontiers** of computational biology, and that's what all the advanced topics and practical assignments are really about. We'd actually like to give you a glimpse of how research works, expose you to current research directions, guide you to find the problems most interesting to you, and help you become an active practicioner in the field. This is achieved through guest lectures, problem sets, labs, and most importantly a term-long independent research **project**, where you carry out your independent research.

The **modules** of the course follow that pattern, each consisting of lectures that cover the foundations and the frontiers of each topic. The foundation lectures introduce the classical problems in the field. These problems are very well understood and elengant solutions have already been found; some have even been taught for well over a decade. The frontiers portion of the module cover advanced topics, usually by tackling central questions that still remain open in the field. These chapters frequently include guest lectures by some of the pioneers in each area speaking both about the general state of the field as well as their own lab's research.

The **assignments** for the course follow the same foundation/frontiers pattern. Half of the assignments are going to be about working out the methods with pencil on paper, and diving deep into the algorithmic and machine learning notions of the problems. The other half are actually going to be practical questions consisting of programming assignments, where real data sets are provided. You will analyze this data using the techniques you have learned and interpret your results, giving you a real hands on experience. The assignments build up to the final project, where you will propose and carry out an original research project, and present your findings in conference format. Overall, the assignments are designed to give you the opportunity to apply computational biology methods to real problems in biology.

1.1.3 Duality of disciplines: Computation and Biology

In addition to aiming to cover both foundations and frontiers, the other important duality of this course is between computation and biology.

From the **biological** perspective of the course, we aim to teach topics that are fundamental to our understanding of biology, medicine, and human health. We therefore shy away from any computationally-interesting problems that are biologically-inspired, but not relevant to biology. We're not just going to see something in biology, get inspired, and then go off into computer science and do a lot of stuff that biology will never care about. Instead, our goal is to work on problems that can make a significant change in the field of biology. We'd like you to publish papers that actually matter to the biological community and have real biological impact. This goal has therefore guided the selection of topics for the course, and each chapter focuses on a fundamental biological problem.

From the **computational** perspective of the course, being after all a computer science class, we focus on exploring general techniques and principles that are certainly important in computational biology, but nonetheless can be applied in any other fields that require data analysis and interpretation. Hence, if what you want is to go into cosmology, meteorology, geology, or any such, this class offers computational techniques that will likely become useful when dealing with real-world data sets related to those fields.

1.1.4 Why Computational Biology?

lecture1_transcript.html#Motivations

There are many reasons why Computational Biology has emerged as an important discipline in recent years, and perhaps some of these lead you to pick up this book or register for this class. Even though we have our own opinion on what these reasons are, we have asked the students year after year for their own view on what has enabled the field of Computational Biology to expand so rapidly in the last few years. Their responses fall into several broad themes, which we sumarize here.

1. Perhaps the most fundamental reason why computational approaches are so well-suited to the study of biological data is that at their core, biological sustems are **fundamentally digital in nature**. To be

blunt, humans are not the first to build a digital computer – our ancestors *are* the first digital computer, as the earliest DNA-based life forms were already storing, copying, and processing digital information encoded in the letters A,C,G, and T. The major evolutionary advantage of a digital medium for storing genetic information is that it can persist across thousands of generations, while analog signals would be diluted from generation to generation from basic chemical diffusion.

2. Besides DNA, many other aspects of biology are digital, such as **biological switches**, which ensure that only two discrete possible states are achieved by feedback loops and metastable processes, even though these are implemented by levels of molecules. Extensive feedback loops and other diverse regulatory circuits implement discrete decisions through otherwise unstable components, again with design principles similar to engineering practice, making our quest to understand biological systems from an engineering perspective more approachable.

3. Sciences that heavily benefit from data processing, such as Computational Biology, follow a virtuous cycle involving the data available for processing. The more that can be done by processing and analyzing the available data, the more funding will be directed into developing technologies to obtain, process and analyze even more data. New technologies such as sequencing, and high-throughput experimental techniques like microarray, yeast two-hybrid, and ChIP-chip assays are creating **enormous and increasing amounts of data** that can be analyzed and processed using computational techniques. The $1000 and $100 genome projects are evidence of this cycle. Over ten years ago, when these projects started, it would have been ludicrous to even imagine processing such massive amounts of data. However, as more potential advantages were devised from the processing of this data, more funding was dedicated into developing technologies that would make these projects feasible.

4. The ability to process data has greatly improved in the recent years, owing to: 1) the massive computational power available today (due to Moore's law, among other things), and 2) the advances in the algorithmic techniques at hand.

5. Optimization approaches can be used to solve, via computational techniques, that are otherwise intractable problems.

6. **Running time & memory** considerations are critical when dealing with huge datasets. An algorithm that works well on a small genome (for example, a bacteria) might be too time or space inefficient to be applied to 1000 mammalian genomes. Also, combinatorial questions dramatically increase algorithmic complexity.

7. Biological datasets can be **noisy**, and filtering signal from noise is a computational problem.

8. **Machine learning** approaches are useful to make inferences, classify biological features, & identify robust signals.

9. As our understanding of biological systems deepens, we have started to realize ethat ach such system cannot be analyzed in isolation. These systems have proved to be intertwined in ways previously unheard of, and we have started to shift our analyses to techniques that consider them all as a whole.

10. It is possible to use computational approaches to find correlations in an unbiased way, and to come up with conclusions that transform biological knowledge and facilitate active learning. This approach is called **data-driven discovery**.

11. Computational studies can **predict** hypotheses, mechanisms, and theories to explain experimental observations. These falsifiable hypotheses can then be tested experimentally.

12. Computational approaches can be used not only to analyze existing data but also to **motivate data collection** andsuggest useful experiments. Also, computational filtering can narrow the experimental search space to allow more focused and efficient experimental designs.

13. Biology has **rules**: Evolution is driven by two simple rules: 1) random mutation, and 2) brutal selection. Biological systems are contrained to these rules, and when analyzing data, we are looking to find and interpret the emerging behavior that these rules generate.

14. **Datasets can be combined** using computational approaches, so that information collected across multiple experiments and using diverse experimental approaches can be brought to bear on questions of interest.

15. Effective **visualizations** of biological data can facilitate discovery.

16. Computational approaches can be used to **simulate & model** biological data.

17. Large scale, systems engineering approaches are facilitated by computational technique to obtain global views into the organism that are too complex to analyze otherwise.

1.1.5 Finding Functional Elements: A Computational Biology Question

lecture1_transcript.html#Codons

Several computational biology problems refer to finding biological signals in DNA data (e.g. coding regions, promoters, enhancers, regulators, ...).

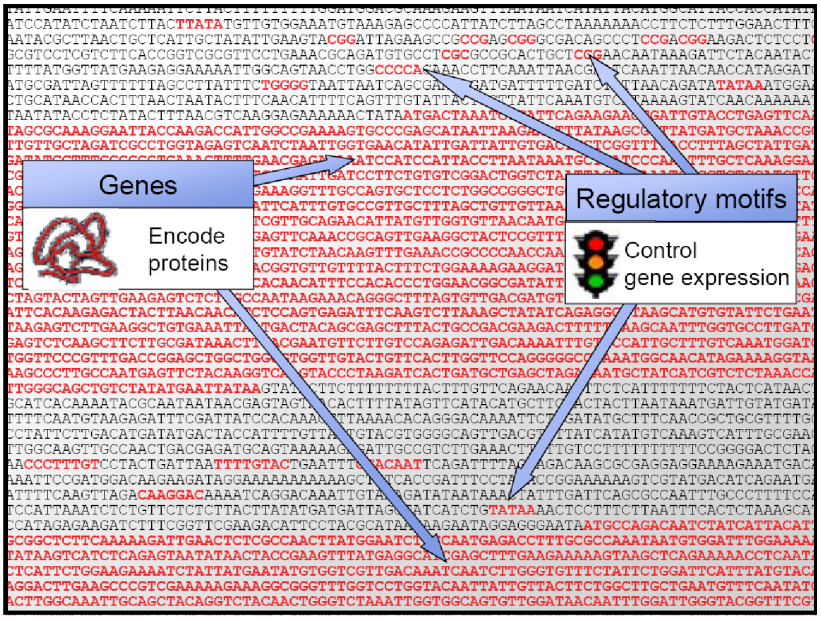


Figure 1.1: In this computational biology problem, we are provided with a sequence of bases, and wish to locate genes and regulatory motifs.

We then discussed a specific question that computational biology can be used to address: how can one find functional elements in a genomic sequence? The slide that is filled with letters shows part of the sequence of the yeast genome. Given this sequence, we can ask:

Q: What are the genes that encode proteins?

A: During translation, the start codon marks the first amino acid in a protein, and the stop codon indicates the end of the protein. However, as indicated in the Extracting signal from noise slide, only a few of these ATG sequences in DNA actually mark the start of a gene which will be expressed as protein. The others are noise; for example, they may have been part of introns (non-coding sequences which are spliced out after transcription).

Q: How can we find features (genes, regulatory motifs, and other functional elements) in the genomic sequence?

A: These questions could be addressed either experimentally or computationally. An experimental approach to the problem would be creating a knockout, and seeing if the fitness of the organism is affected. We

could also address the question computationally by seeing whether the sequence is conserved across the genomes of multiple species. If the sequence is significantly conserved across evolutionary time, its likely to perform an important function.

There are caveats to both of these approaches. Removing the element may not reveal its functioneven if there is no apparent difference from the original, this could be simply because the right conditions have not been tested. Also, simply because an element is not conserved doesnt mean it isnt functional. (Also, note that functional element is an ambiguous term. Certainly, there are many types of functional elements in the genome that are not protein-encoding. Intriguingly, 90-95% of the human genome is transcribed (used as a template to make RNA). It isnt known what the function of most of these transcribed regions are, or indeed if they are functional).

1.2 Final Project - Introduction to Research In Computational Biology

lecture1_transcript.html#FinalProject

1.2.1 Final project goals

An important component of being a computational biologist is the ability to carry out independent research in the area. The skills for a successful researcher differ from one person to the next, but in the process of teaching this course, we have identified several aspects that are all needed, and laid out activities for a term-long project, that enable students to carry out their independent research.

The project mirrors real world scientific process: come up with an idea → frame it → propose it→ revise it → carry it out → present your results. Students are expected to think critically about their own project, and also evaluate peer research proposals, and lastly respond to feedback from their peers.

Students are expected to use real data and present their results in conference format. The ultimate goal is publishable research. Students are encouraged to talk with the course staff while formulating a final project idea, look head through the various chapters and modules, and get an idea of what areas will interest you most.

1.2.2 Final project milestones

Instead of waiting until the end of the term to begin brainstorming or provide feedback, we begin project activities with the first problem set, to identify problems of interest and types of projects, find partners, speak with current students and postdocs in computational biology that can serve as mentors, and lay out a research plan in the style of an NIH proposal to identify potential pitfalls early and address them or work around them before they become a bottleneck.

By setting up several incremental progress milestones throughout the term, coupled with mentoring and feedback throughout the semester, we have achieved consistent progress in previous years, which can be useful to students taking on a new project at any stage of their career. Research projects from this course in the past have been used as the starting point for a published paper, have led to Masters and PhD theses, and earned awards both academically and in conferences.

The timeline for final project is as follows:

1. **Set-up:** a brief overview of your experience and interest. Due 9/26
2. **Brainstorming:** a list of initial project ideas and partners. Due 10/12
3. **Proposal:** submit a project proposal in the form of an NIH proposal. Due 10/24
4. **Review:** review and critique 3 peer proposals. Due 10/31
5. **Midterm Progress Report:** write outline of final report. Due 11/28
6. **Final Project Report:** write report in conference paper format. Due 12/9

7. **Final Class Presentation:** 10min conference talk. Due 12/13

There will be Friday mentoring sessions before each portion of the final project is due, and you are encouraged to find a mentor at the first few sessions who is actively interested in your project and could help you more frequently. The mentoring sessions can be helpful in identifying if unexpected results are the result of a bug or are instead a discovery.

Make sure you start working on the project even while waiting for peer reviews, so that you will have 4-5 weeks to complete the research itself.

1.2.3 Project deliverables

The final project will include the following two deliverables:

1. A written presentation, due Mon at 8pm, last week of classes. The written presentation can contain the following elements:
   - Who did what (to reflect trend in publications)
   - The overall project experience
   - Your discoveries
   - What you learned from the experience (introspection)
2. An oral presentation, due Thursday after the written presentation. This allows students three days to prepare the oral presentation.

1.2.4 Project grading

Selecting a project that will be successful can be difficult. To help students optimize for a successful project, we let them know in advance the grading scheme, designed to maximize the project impact by being original, challenging, and relevant to the field, but of course the grade is ultimately dependent on the overall achievement and the clarity of presentation.

Briefly, the grading equation for the final project is:
$min(O, C, R)xA + P$
where

Originality - unoriginal computational experiments don't get published

Challenge - the project needs to be sufficiently difficult

Relevance - it needs to be from biology, can't just reuse something from another field

Achievement - if you don't accomplish anything you won't get a good grade

Presentation - even if you've achieved a good project you have to be able to present it so everyone knows that, and make it look easy. the presentation should show how the project is O, C, and R.

Originality, Challenge, Relevance are each out of 5 points, Achievement and Presentation are each out of 10.

1.3 Additional materials

1.3.1 Online Materials for Fall 2011

lecture1_transcript.html#Handouts

In addition to these *static* notes, the course has several online resources:

- The course website can be accessed at: http://compbio.mit.edu/6.047 , where course material and information from each term can be found

- A wiki http://6047.wikispaces.com, where students sign up for scribing each lecture
- The course calendar on Google Calendar. You can add "6.047 Lectures", a public calendar.
- The NB note-taking system for annotating these notes http://nb.mit.edu/
- Lastly, videos and audio recordings of all lectures from Fall 2010 are freely available online at http://compbio.mit.edu/teaching.html#compbioF10

1.3.2 Textbooks

lecture1_transcript.html#CourseInformation The following three (optional) reference textbooks recommended for the class.

1. Richard Durbin, Sean R. Eddy, Anders Krogh and Graeme Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.
2. Neil Jones and Pavel Pevzner, An Introduction to Bioinformatics Algorithms.
3. Richard Duda, Peter Hart, David Stork, Pattern Classification.

Each book has a different advantage. The first book is a classic one. It is heavy in math and covers much of what is in class. The book is focused on sequence alignment. As part of sequence alignment theory, the book aproaches Hidden Markov Models (HMM), pairwise and multiple alignment methods, phylogenetic trees as well as a short background in probability theory.

The second book intendes to balance between mathematical rigorous and biological relevance. According to the author, it is a good book for undergrad students. The book includes a table that assossiates algorithm to biological problems.

The third book is about machine learning. It uses a more engineering approach. It includes machine learning theory, neural network and, as the name sugests, pattern recognition.

1.4 Crash Course in Molecular Biology

For the primarily computational students, we provide a brief introduction to the key notions of molecular biology that we will encounter throughout the term.

1.4.1 The Central Dogma of Molecular Biology

lecture1_transcript.html#CentralDogma $DNA \rightarrow RNA \rightarrow Protein$

The central dogma of molecular biology describes how information is stored and used in the cell: The genetic code of an organism is stored in DNA, which is transcribed into RNA, which is then translated into protein, that carries out most processes in the cell.

While the central dogma generally holds true, there are many exceptions to this unidirectional flow of information, such as reverse-transcription from RNA to DNA which is used by retroviruses.

Did You Know?

The central dogma is sometimes **incorrectly** interpreted too strongly as meaning that DNA only stores immutable information from one generation to the next that remains identical within a generation, RNA is only used as a temporary information transfer medium, and proteins are the only molecule that can carry out complex actions.
Again, there are many exceptions to this interpretation, for example:

- Somatic mutations can alter the DNA within a generation, and different cells can have different DNA content.
- Some cells undergo programmed DNA alterations during maturation, resulting in different DNA content, most famously the B and T immunity while blood cells
- Epigenetic modifications of the DNA can be inherited from one generation to the next
- RNA can play many diverse roles in gene regulation, metabolic sensing, and enzymatic reactions, functions that were previously thought to be reserved to proteins.
- Proteins themselves can undergo conformational changes that are epigenetically inherited notably prion states that were famously responsible for mad cow disease

1.4.2 DNA

DNA $\rightarrow RNA \rightarrow Protein$

The genetic code of an organism is contained within the DNA molecule. The DNA molecule contains biological signals, some of which encode proteins and some of which encode regulators, signals that turn genes on and off, or decide whether a gene should be turned on or off. Within the genetic code of DNA lies both the data about the proteins that need to be encoded, and the control circuitry, in the form of regulatory motifs.

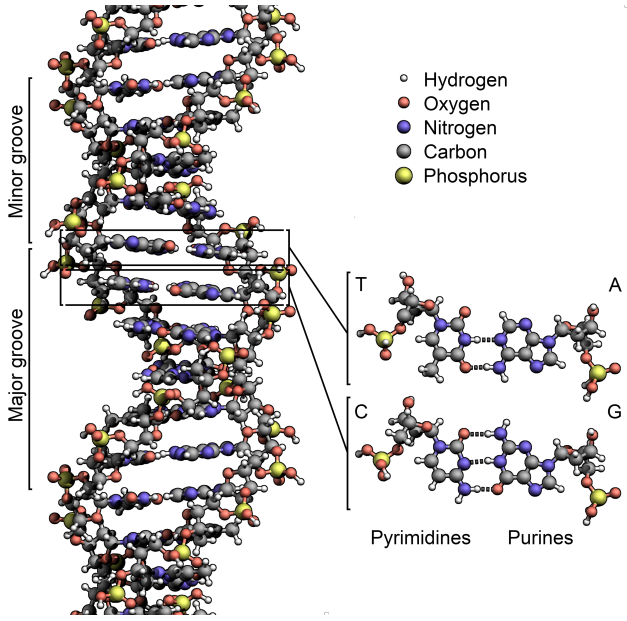


Figure 1.2: The double-helix structure of DNA. Nucleotides are in the center, and the sugar-phosphate backbone lies on the outside.

TODO: Figure out licensing from
http://en.wikipedia.org/wiki/File:DNA_Structure%2BKey%2BLabelled.pn_NoBB.png *@scribe:*

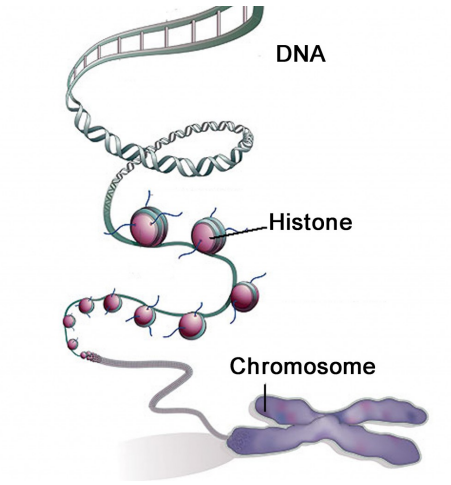


Figure 1.3: DNA is packed over several layers of organization into a compact chromosome.

DNA is the molecular of heredity. DNA is a double helix composed of two complementary strands, with a backbone of phosphate and sugar (deoxyribose) on the outside and bases in the center. Complementary bases are held together by hydrogen bonds. (lecture1_transcript.html#WatsonCrick)

DNA is composed of four **nucleotides**: A(**adenine**), C(**cytosine**),T (**thymine**), and G (**guanine**). A and G are purines, which have two rings, while C and T are pyrimidines, with one ring. A and T are connected by two hydrogen bonds, while C and G are connected by three bonds. Therefore, the A-T pairing is weaker than the C-G pairing. (For this reason, the genetic composition of bacteria that live in hot springs is 80% G-C). lecture1_transcript.html#Complementarity

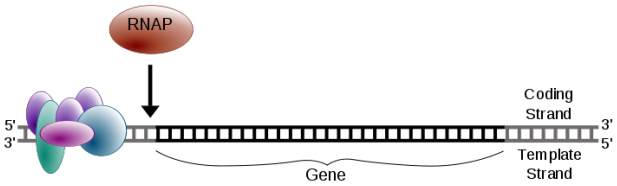
The two DNA strands in the double helix are **complementary**, meaning that if there is an A on one strand, it will be bonded to a T on the other, and if there is a C on one strand, it will be bonded to a G on the other. The DNA strands also have **directionality**, which refers to the positions of the pentose ring where the phosphate backbone connects. This directionality convention comes from the fact that DNA and RNA polymerase synthesize in the 5' to 3' direction. With this in mind, we can say that that the DNA strands are **anti-parallel**, as the 5' end of one strand is adjacent to the 3' end of the other. As a result, DNA can be read both in the 3' to 5' direction and the 5' to 3' direction, and genes and other functional elements can be found in each. By convention, DNA is written from 5' to 3'. The 5' and 3' directions refer to the positions on the pentose ring where the phosphate backbone connects.

The structure of DNA, with its weak hydrogen bonds between the bases in the center, allows the strands to easily be separated for the purpose of DNA replication (the capacity for DNA strands to be separated also allows for transcription, translation, recombination, and DNA repair, among others). This was noted by Watson and Crick as It has not escaped our notice that the specific pairing that we have postulated immediately suggests a possible copying mechanism for the genetic material. In the replication of DNA, the two complementary strands are separated, and each of the strands are used as templates for the construction of a new strand.

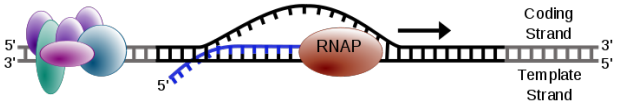
DNA polymerases attach to each of the strands at the origin of replication, reading each existing strand from the 3 to 5 direction and placing down complementary bases such that the new strand grows in the 5 to 3 direction. Because the new strand must grow from 5 to 3, one strand (the leading strand) can be copied continuously, while the other (the lagging strand) grows in pieces which are later glued together by DNA ligase. The end result is 2 double-stranded pieces of DNA, where each is composed of 1 old strand, and 1 new strand; for this reason, DNA replication is semiconservative.

DNA needs to be packed tightly, as it is very long, and the cell is small. The DNA molecule is coiled around histone proteins, forming nucleosomes. These are in turn coiled into chromatin fiber. In Eukaryotes, the chromatin is further packaged as chromosomes. The chromosomes are in turn located in the nucleus. lecture1_transcript.html#Structure

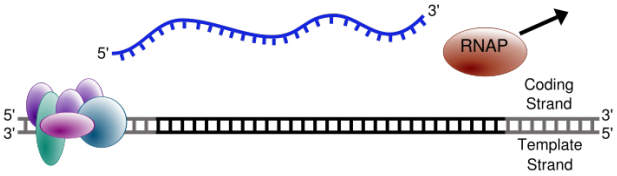
Many organisms have their DNA broken into several chromosomes. Each chromosome contains two



(a) Transcription initiation



(b) Transcription elongation



(c) Transcription termination

Figure 1.4: RNA is produced from a DNA template during transcription. A bubble is opened in the DNA, allowing the RNA polymerase to enter and place down bases complementary to the DNA.

strands of DNA, which are complementary to each other but are read in opposite directions. Genes can occur on either strand of DNA. The DNA before a gene (in the 5' region) is considered "upstream" whereas the DNA after a gene (in the 3' region) is considered "downstream".

Before DNA can be replicated or transcribed into RNA, it must be locally unpacked. Thus, gene expression may be regulated by modifications to the chromatin structure, which make it easier or harder for the DNA to be unpacked. This regulation of gene expression via chromatin modification is an example of epigenetics.

1.4.3 Transcription

lecture1_transcript.html#Transcription

$DNA \rightarrow RNA \rightarrow Protein$

Transcription is the process by which RNA is produced using a DNA template. The DNA is partially unwound to form a bubble, and RNA polymerase attaches to the promoter region of the template strand, reading it from the 3 to 5 direction and placing down complementary bases (except U is used instead of T). The newly produced RNA is grown from the 5 to 3 direction.

In eukaryotes, following transcription, among the modifications that are made to pre-mRNA include splicing out the **introns**, intervening regions which dont code for protein, so that only the coding regions, the **exons**, remain. Different regions of the primary transcript may be spliced out to lead to different protein products (alternative splicing).

1.4.4 RNA

lecture1_transcript.html#RNA

$DNA \rightarrow RNA \rightarrow Protein$

RNA is produced when DNA is transcribed. It is structurally similar to DNA, with the following major differences:

1. The nucleotide uracil (U) is used instead of DNA's thymine (T).

2. RNA contains ribose instead of deoxyribose (deoxyribose lacks the oxygen molecule on the 2 position found in ribose).

3. RNA is single-stranded, whereas DNA is double-stranded.

RNA molecules are the intermediary step to code a protein. RNA molecules also have catalytic and regulatory functions. One example of catalytic function is in protein synthesis, where RNA is part of the ribosome.

There are many different types of RNA, including:

1. **mRNA** (messenger RNA) contains the information to make a protein and is translated into protein sequence.

2. **tRNA** (transfer RNA) specifies codon-to-amino-acid translation. It contains a 3 base pair anti-codon complementary to a codon on the mRNA, and carries the amino acid corresponding to its anticodon attached to its 3 end.

3. **rRNA** (ribosomal RBA) forms the core of the ribosome, the organelle responsible for the translation of mRNA to protein.

4. **snRNA** (small nuclear RNA) is involved in splicing (removing introns from) pre- mRNA, as well as other functions.

Other functional kinds of RNA exist and are still being discovered. RNA molecules can have complex three-dimensional structures and perform diverse functions in the cell.

According to the "RNA world" hypothesis, early life was based entirely on RNA. RNA served as both the information repository (like DNA today) and the functional workhorse (like protein today) in early organisms. Protein is thought to have arisen afterwards via ribosomes, and DNA is thought to have arisen last, via reverse transcription.

1.4.5 Translation

lecture1_transcript.html#Translation

$DNA \rightarrow RNA \rightarrow Protein$

Unlike transcription, in which the nucleotides remained the means of encoding information in both DNA and RNA, when RNA is translated into protein, the primary structure of the protein is determined by the sequence of amino acids of which it is composed. Since there are 20 amino acids and only 4 nucleotides, 3-nucleotides sequences in mRNA, known as codons, encode for each of the 20 amino acids.

Each of the 64 possible 3-sequences of nucleotides (codon) uniquely specifies either a particular amino acid, or is a stop codon that terminates protein translation (the start codon also encodes methionine). Since there are 64 possible codon sequences, the code is degenerate, and some amino acids are specified by multiple encodings. Most of the degeneracy occurs in the 3rd codon position.

1.4.6 Protein

$DNA \rightarrow RNA \rightarrow Protein$

Protein is the molecule responsible for carrying out most of the tasks of the cell, and can have many functions, such as enzymatic, contractile, transport, immune system, signal and receptor to name a few. Like RNA and DNA, proteins are polymers made from repetitive subunits. Instead of nucleotides, however, proteins are composed of amino acids.

Each amino acid has special properties of size, charge, shape, and acidity. As such, additional structure emerges beyond simply the sequence of amino acids (the primary structure), as a result of interactions between the amino acids. As such, the three-dimensional shape, and thus the function, of a protein is determined by its sequence. However, determining the shape of a protein from its sequence is an unsolved problem in computational biology.

SECOND POSITION

FIRST POSITION	U	C	A	G	THIRD POSITION
U	phenyl-alanine	serine	tyrosine	cysteine	U
					C
	leucine		stop	stop	A
			stop	tryptophan	G
C	leucine	proline	histidine	arginine	U
					C
			glutamine		A
					G
A	isoleucine	threonine	asparagine	serine	U
					C
	* methionine		lysine	arginine	A
					G
G	valine	alanine	aspartic acid	glycine	U
					C
			glutamic acid		A
					G

* and start

Figure 1.5: This codon table shows which of the 20 amino acid each of the 3-nucleotide codons in mRNA are translated into. In red are the stop codons, which terminate translation.

1.4.7 Regulation: from Molecules to Life

lecture1_transcript.html#Regulation

Not all genes are expressed at the same time in a cell. For example, cells would waste energy if they produced lactose transporter in the absence of lactose. It is important for a cell to know which genes it should expresses and when. A regulatory network is involved to control expression level of genes in a specific circumstance.

Transcription is one of the steps at which protein levels can be regulated. The promoter region, a segment of DNA found upstream (past the 5 end) of genes, functions in transcriptional regulation. The promoter region contains motifs that are recognized by proteins called transcription factors. When bound, transcription factors can recruit RNA polymerase, leading to gene transcription. However, transcription factors can also participate in complex regulatory interactions. There can be multiple binding sites in a promotor, which can act as a logic gate for gene activation. Regulation in eukaryokes can be extremely complex, with gene expression affected not only by the nearby promoter region, but also by distant enhancers and repressors.

We can use probabilistic models to identify genes that are regulated by a given transcription factor. For example, given the set of motifs known to bind a given transcription factor, we can compute the probability that a candidate motif also binds the transcription factor (see the notes for precept #1). Comparative sequence analysis can also be used to identify regulatory motifs, since regulatory motifs show characteristic patterns of evolutionary conservation.

The lac operon in E. coli and other bacteria is an example of a simple regulatory circuit. In bacteria, genes with related functions are often located next to each other, controlled by the same regulatory region, and transcribed together; this group of genes is called an operon. The lac operon functions in the metabolism of the sugar lactose, which can be used as an energy source. However, the bacteria prefer to use glucose as an energy source, so if there is glucose present in the environment the bacteria do not want to make the proteins that are encoded by the lac operon. Therefore, transcription of the lac operon is regulated by an elegant circuit in which transcription occurs only if there is lactose but not glucose present in the environment.

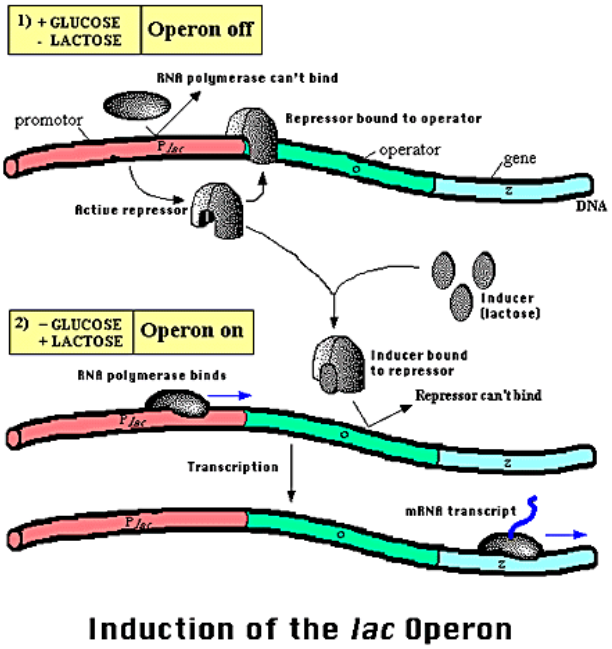


Figure 1.6: Operon Lac illustrates a simple biological regulatory system. In the presence of glucose, genes to lactose metabolism are turn out because glucose inactives an activator protein. In the absence of lactose, a repressor protein also turns out the operon. Lactose metabolism genes are expressed only in the presence of lactose and absence of glucose.

1.4.8 Modules in Bio-Network

lecture1_transcript.html#Networks

Several genes work together in order to accomplish a specific task. A set of genes which work together is called module. Modules are useful tool to understand the main mechanism responsible to a biological condition. Computational biology tools can be used to find modules as well as use modules and make predictions.

1.4.9 Metabolism

lecture1_transcript.html#

Live organisms are made from self-organizing building blocks. Energy source is necessary for organize blocks. The basic mechanism involved in building blocks is degrading small molecules to get energy to build big molecules. The process of degrading molecules to release energy is called catabolism and the process of using energy to assemble more complex molecules is called anabolism. Anabolism and catabolism are both metabolic processes. Metabolism regulates the flow of mass and energy in order to keep an organism in a state of low entropy.

Enzymes are a critical component of metabolic reactions. The vast majority of (but not all!) enzymes are proteins. Many biologically critical reactions have high activation energies, so that the uncatalyzed reaction would happen extremely slowly or not at all. Enzymes speed up these reactions, so that they can happen at a rate that is sustainable for the cell. In living cells, reactions are organized into metabolic pathways. A reaction may have many steps, with the products of one step serving as the substrate for the next. Also, metabolic reactions often require an investment of energy (notably as a molecule called ATP), and energy released by one reaction may be captured by a later reaction in the pathway. Metabolic pathways are also important for the regulation of metabolic reactionsif any step is inhibited, subsequent steps may lack the substrate or the energy that they need to proceed. Often, regulatory checkpoints appear early in metabolic pathways, since if the reaction needs to be stopped, it is obviously better to stop it before much energy has

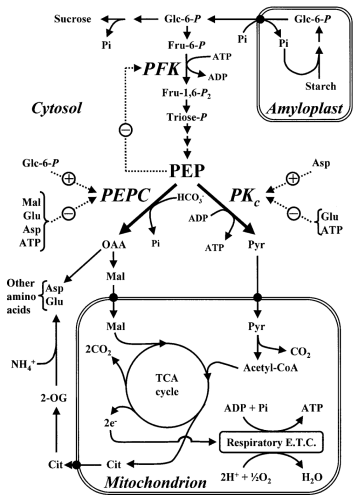


Figure 1.7: Metabolic pathways and regulation can be studied by Computational biology. Models are made from genome scale information and used to predict metabolic function and to metabolic engineering. An example of biological engineering is modifying bacteria genome to overproduce artemesenin, an antibiotic used to treat malaria.

been invested.

1.4.10 Systems Biology

lecture1_transcript.html#SystemsBiology

Systems biology strives to explore and explain the behavior that emerges from the complex interactions among the components of a biological system. One interesting recent paper in systems biology is Metabolic gene regulation in a dynamically changing environment (Bennett et al., 2008). This work makes the assumption that yeast is a linear, time invariant system, and runs a signal (glucose) through the system to observe the response. A periodic response to low-frequency fluctuations in glucose level is observed, but there is little response to high-frequency fluctuations in glucose level. Thus, this study finds that yeast acts as a low-pass filter for fluctuations in glucose level.

1.4.11 Synthetic Biology

lecture1_transcript.html#SyntheticBiology

Not only can we use computational approaches to model and analyze biological data collected from cells, but we can also design cells that implement specific logic circuits to carry out novel functions. The task of designing novel biological systems is known as synthetic biology.

A particularly notable success of synthetic biology is the improvement of artemesenin production. Artemesenin is a drug used to treat malaria. However, artemisinin was quite expensive to produce. Recently, a strain of yeast has been engineered to synthesize a precursor to artemisinic acid at half of the previous cost.

1.4.12 Model organisms and human biology

Diverse model organisms exist for all aspects of human biology. Importance of using model organisms at appropriate level of complexity.

Note: In this particular book, we'll focus on human biology, and we'll use examples from baker's yeast *Saccharomyces cerevisiae*, the fruitfly *Drosophila melanogaster*, the nematode worm *Coenorhabditis elegans*, and the house mouse *Mus musculus*. We'll deal with bacterial evolution only in the context of metagenomics of the human microbiome.

1.5 Introduction to algorithms and probabilistic inference

1. We will quickly review some basic probability by considering an alternate way to represent motifs: a *position weight matrix* (PWM). We would like to model the fact that proteins may bind to motifs that are not fully specified. That is, some positions may require a certain nucleotide (e.g. `A`), while others positions are free to be a subset of the 4 nucleotides (e.g. `A` or `C`). A PWM represents the set of all DNA sequences that belong to the motif by using a matrix that stores the probability of finding each of the 4 nucleotides in each position in the motif. For example, consider the following PWM for a motif with length 4:

	1	2	3	4
A	0.6	0.25	0.10	1.0
G	0.4	0.25	0.10	0.0
T	0.0	0.25	0.40	0.0
C	0.0	0.25	0.40	0.0

   We say that this motif can generate sequences of length 4. PWMs typically assume that the distribution of one position is not influenced by the base of another position. Notice that each position is associated with a probability distribution over the nucleotides (they sum to 1 and are nonnegative).

2. We can also model the *background distribution* of nucleotides (the distribution found across the genome):

A	0.1
G	0.4
T	0.1
C	0.4

   Notice how the probabilities for A and T are the same and the probabilities of G and C are the same. This is a consequence of the complementarity DNA which ensures that the overall composition of A and T, G and C is the same overall in the genome.

3. Consider the sequence $S =$ GCAA.

   The probability of the motif generating this sequence is $P(S|M) = 0.4 \times 0.25 \times 0.1 \times 1.0 = 0.01$.

   The probability of the background generating this sequence $P(S|B) = 0.4 \times 0.4 \times 0.1 \times 0.1 = 0.0016$.

4. Alone this isn't particularly interesting. However, given fraction of sequences that are generated by the motif, e.g. $P(M) = 0.1$, and assuming all other sequences are generated by the background ($P(B) = 0.9$) we can compute the probability that the motif generated the sequence using Bayes' Rule:

$$\begin{aligned} P(M|S) &= \frac{P(S|M)P(M)}{P(S)} \\ &= \frac{P(S|M)P(M)}{P(S|B)P(B) + P(S|M)P(M)} \\ &= \frac{0.01 \times 0.1}{0.0016 \times 0.9 + 0.01 \times 0.1} = 0.40984 \end{aligned}$$

1.5.1 Probability distributions

1.5.2 Graphical probabilistic models

1.5.3 Bayes rules: priors, likelihood, posterior

1.5.4 Markov Chains and Sequential Models

1.5.5 Probabilistic inference and learning

1.5.6 Max Likelihood and Max A Posteriori Estimates

Additional materials

Online Materials for Fall 2011

lecture1_transcript.html#Handouts

In addition to these *static* notes, the course has several online resources:

- The course website can be accessed at: http://compbio.mit.edu/6.047 , where course material and information from each term can be found
- A wiki http://6047.wikispaces.com, where students sign up for scribing each lecture
- The course calendar on Google Calendar. You can add "6.047 Lectures", a public calendar.
- The NB note-taking system for annotating these notes http://nb.mit.edu/
- Lastly, videos and audio recordings of all lectures from Fall 2010 are freely available online at http://compbio.mit.edu/teaching.html#compbioF10

Textbooks

lecture1_transcript.html#CourseInformation The following three (optional) reference textbooks recommended for the class.

1. Richard Durbin, Sean R. Eddy, Anders Krogh and Graeme Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.
2. Neil Jones and Pavel Pevzner, An Introduction to Bioinformatics Algorithms.
3. Richard Duda, Peter Hart, David Stork, Pattern Classification.

Each book has a different advantage. The first book is a classic one. It is heavy in math and covers much of what is in class. The book is focused on sequence alignment. As part of sequence alignment theory, the book aproaches Hidden Markov Models (HMM), pairwise and multiple alignment methods, phylogenetic trees as well as a short background in probability theory.

The second book intendes to balance between mathematical rigorous and biological relevance. According to the author, it is a good book for undergrad students. The book includes a table that assossiates algorithm to biological problems.

The third book is about machine learning. It uses a more engineering approach. It includes machine learning theory, neural network and, as the name sugests, pattern recognition.

Bibliography

[1] lec1test. lec1test, lec1test.

Part I

Comparing Genomes

CHAPTER

TWO

SEQUENCE ALIGNMENT AND DYNAMIC PROGRAMMING

Guilherme Issao Fuijwara, Pete Kruskal (2007)
Arkajit Dey, Carlos Pards (2008)
Victor Costan, Marten van Dijk (2009)
Andreea Bodnari, Wes Brown (2010)
Sarah Spencer (2011)
Nathaniel Parrish (2012)

Figures

2.1 Introduction

Evolution has preserved functional elements in the genome. Such preserved elements between species are often homologs[1] – either orthologous or paralogous sequences (refer to Appendix 2.11.1). Orthologous gene sequences are of higher interest in the study of evolutionary paths due to the higher influence of purifying selection[2], since such regions are extremely well preserved. A common approach to the analysis of evolutionary similarities is the method of aligning sequences, primarily solved using computational methods (e.g., dynamic programming). These notes discuss the sequence alignment problem, the technique of dynamic programming, and a specific solution to the problem using this technique.

[1] Homologous sequences are genomic sequences descended from a common ancestor.

[2] In the human genome, only $\approx 5\%$ of the nucleotides are under selection. Appendix 2.11.2 has more details on types of selection.

2.2 Aligning Sequences

Sequence alignment represents the method of comparing two or more genetic strands, such as DNA or RNA. These comparisons help with the discovery of genetic commonalities and with the (implicit) tracing of strand evolution. There are two main types of alignment:

- Global alignment: an attempt to align every element in a genetic strand, most useful when the genetic strands under consideration are of roughly equal size. Global alignment can also end in gaps.
- Local alignment: an attempt to align regions of sequences that contain similar sequence motifs within a larger context.

2.2.1 Example Alignment

Within orthologous gene sequences, there are islands of conservation, or relatively large stretches of nucleotides that are preserved between generations. These conserved regions typically imply functional elements and vice versa. As an example, we considered the alignment of the Gal10-Gal1 intergenic region for four different yeast species, the first cross-species whole genome alignment (Figure 2.1). As we look at this alignment, we note that some areas are more conserved than others. In particular, we note some small conserved motifs such as CGG and CGC, which in fact are functional elements in the binding of Gal4[7]. This example illustrates how we can read evolution to find functional elements.

We have to be cautious with our interpretations, however, because conservation does sometimes occur by random chance. In order to extract accurate biological information from sequence alignments we have to separate true signatures from noise. The most common approach to this problem involves modeling the evolutionary process. By using known codon substitution frequencies and RNA secondary structure constraints, for example, we can calculate the probability that evolution acted to preserve a biological function. See Chapter **??** for an in-depth discussion of evolutionary modeling and functional conservation in the context of genome annotation.

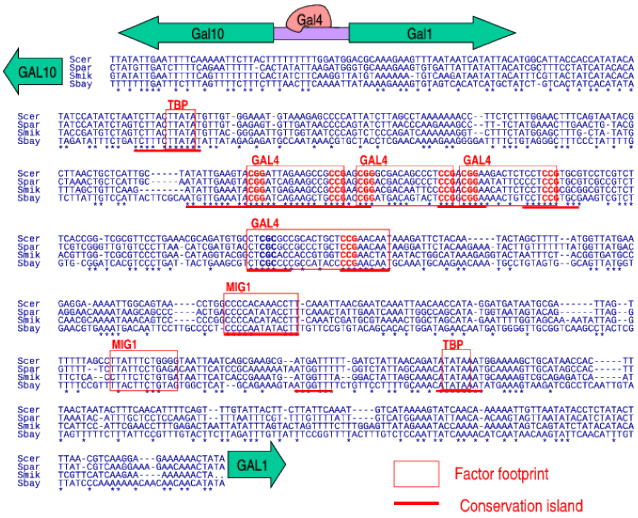


Figure 2.1: Sequence alignment of Gal10-Gal1 between four yeast strains. Asterisks mark conserved nucleotides.

2.2.2 Solving Sequence Alignment

The genome changes over time, and, lacking a time machine, we cannot compare genomes of living species with their ancestors. Thus, we are limited to comparing just the genomes of living descendants. The goal of sequence alignment is to infer the 'edit operations' that change a genome by looking only at these endpoints.

We must make some assumptions when performing sequence alignment, if only because we must transform a biological problem into a computationally feasible one and we require a model with relative simplicity and tractability. In practice, the majority of sequence evolution occurs in the form of nucleotide mutations, deletions, and insertions (Figure 2.2). Thus, our sequence alignment model will only consider these three operations and will ignore other realistic events that occur with lower probability.

1. A nucleotide **mutation** occurs when some nucleotide in a sequence changes to some other nucleotide during the course of evolution.

2. A nucleotide **deletion** occurs when some nucleotide is deleted from a sequence during the course of evolution.

3. A nucleotide **insertion** occurs when some nucleotide is added to a sequence during the course of evolution.

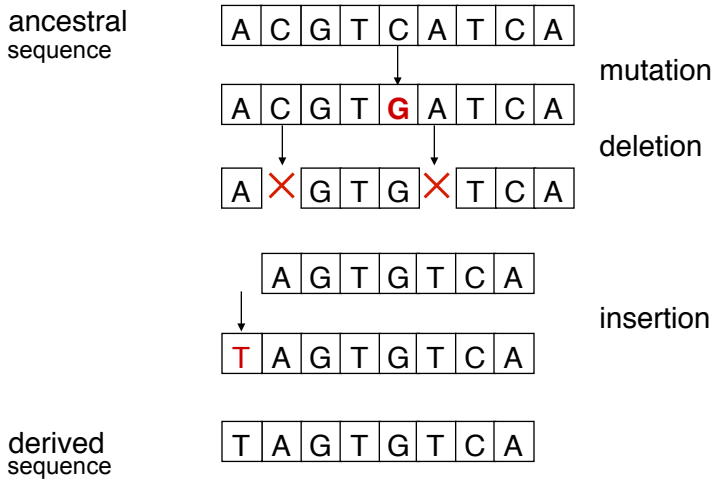


Figure 2.2: Evolutionary changes of a genetic sequence

Note that these three events are all reversible. For example, if a nucleotide N mutates into some nucleotide M, it is also possible that nucleotide M can mutate into nucleotide N. Similarly, if nucleotide N is deleted, the event may be reversed if nucleotide N is (re)inserted. Clearly, an insertion event is reversed by a corresponding deletion event.

This reversibility is part of a larger design assumption: time-reversibility. Specifically, any event in our model is reversible in time. For example, a nucleotide deletion going forward in time may be viewed as a nucleotide insertion going backward in time. This is useful because we will be aligning sequences which both exist in the present. In order to compare evolutionary relatedness, we will think of ourselves following one sequence backwards in time to a common ancestor and then continuing forward in time to the other sequence. In doing so, we can avoid the problem of not having an ancestral nucleotide sequence.

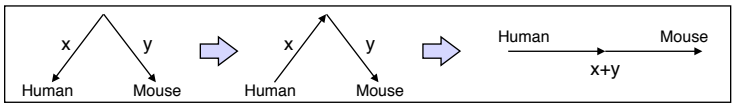


Figure 2.3: Aligning human to mouse sequences is analogous to tracing backward from the human to a common ancestor, then forward to the mouse

Note that time-reversibility is useful in solving some biological problems but does not actually apply to biological systems. For example, CpG (where p denotes the phosphate backbone in a DNA strand) may incorrectly pair with a TpG or CpA during DNA replication, but the reverse operation cannot occur; it is

not time-reversible. To be very clear, time-reversibility is simply a design decision in our model; it is not inherent to the biology[3].

We also need some way to evaluate our alignments. There are many possible sequences of events that could change one genome into another. Perhaps the most obvious ones minimize the number of events (i.e., mutations, insertions, and deletions) between two genomes, but sequences of events in which many insertions are followed by corresponding deletions are also possible. We wish to establish an optimality criterion that allows us to pick the 'best' series of events describing changes between genomes.

We choose to invoke Occam's razor and select a maximum parsimony method as our optimality criterion. That is, in general, we wish to minimize the number of events used to explain the differences between two nucleotide sequences. In practice, we find that point mutations are more likely to occur than insertions and deletions, and certain mutations are more likely than others. Our parsimony method must take these and other inequalities into account when maximizing parsimony. This leads to the idea of a substitution matrix and a gap penalty, which are developed in the following sections. Note that we did not need to choose a maximum parsimony method for our optimality criterion. We could choose a probabilistic method, for example using Hidden Markov Models (HMMs), that would assign a probability measure over the space of possible event paths and use other methods for evaluating alignments (e.g., Bayesian methods). Note the duality between these two approaches; our maximum parsimony method reflects a belief that mutation events have low probability, thus in searching for solutions that minimize the number of events we are implicitly maximizing their likelihood.

2.3 Problem Formulations

In this section, we introduce a simple problem, analyze it, and iteratively increase its complexity until it closely resembles the sequence alignment problem. This section should be viewed as a warm-up for Section 2.5 on the Needleman-Wunsch algorithm.

2.3.1 Formulation 1: Longest Common Substring

As a first attempt, suppose we treat the nucleotide sequences as strings over the alphabet A, C, G, and T. Given two such strings, S1 and S2, we might try to align them by finding the longest common substring between them. In particular, these substrings cannot have gaps in them.

As an example, if S1 = ACGTCATCA and S2 = TAGTGTCA (refer to Figure 2.4), the longest common substring between them is GTCA. So in this formulation, we could align S1 and S2 along their longest common substring, GTCA, to get the most matches. A simple algorithm would be to try aligning S1 with different offsets of S2 and keeping track of the longest substring match found thus far. Note that this algorithm is quadratic in the lengths of S1 and S2, which is slower than we would prefer for such a simple problem.

2.3.2 Formulation 2: Longest Common Subsequence (LCS)

Another formulation is to allow gaps in our subsequences and not just limit ourselves to substrings with no gaps. Given a sequence X = (x_1 ,.., x_m), we formally define $Z = (z_1, \ldots, z_k)$ to be a subsequence of X if there exists a strictly increasing sequence $i_1 < i_2 < \ldots < i_k$ of indices of X such that for all j, $1 \leq j \leq k$, we have $x_{i_j} = z_j$(CLRS 350-1).

In the longest common subsequence (LCS) problem, we're given two sequences X and Y and we want to find the maximum-length common subsequence Z. Consider the example of sequences S1 = ACGTCATCA and S2 = TAGTGTCA (refer to Figure 2.5). The longest common subsequence is AGTTCA, a longer match than just the longest common substring.

[3]This is an example where understanding the biology helps the design greatly, and illustrates the general principle that success in computational biology requires strong knowledge of the foundations of both CS and biology. Warning: computer scientists who ignore biology will work too hard.

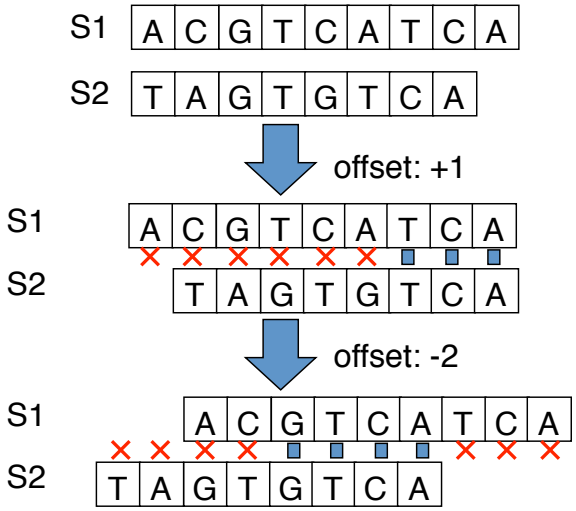


Figure 2.4: Example of longest common substring formulation

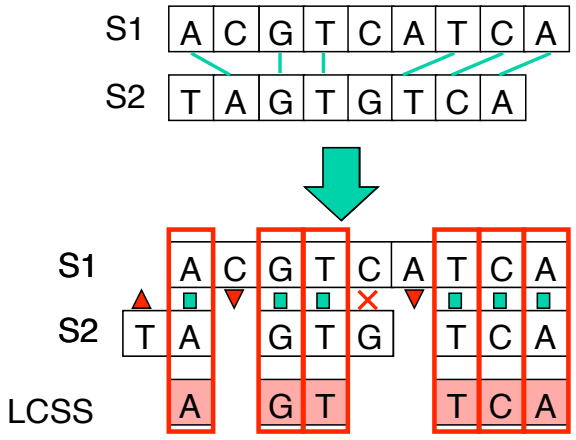


Figure 2.5: Example of longest common subsequence formulation

2.3.3 Formulation 3: Sequence Alignment as Edit Distance

The previous LCS formulation is close to the full sequence alignment problem, but so far we have not specified any cost functions that can differentiate between the three types of edit operations (insertion, deletions, and substitutions). Implicitly, our cost function has been uniform, implying that all operations are equally likely. Since substitutions are much more likely, we want to bias our LCS solution with a cost function that prefers substitutions over insertions and deletions.

We recast sequence alignment as a special case of the classic Edit-Distance[4] problem in computer science (CLRS 366). We add varying penalties for different edit operations to reflect biological occurrences. One biological reasoning for this scoring decision is the probabilities of bases being transcribed incorrectly during polymerization. Of the four nucleotide bases, A and G are purines (larger, two fused rings), while C and T are pyrimidines (smaller, one ring). Thus RNA polymerase[5] is much more likely to confuse two purines or two pyrimidines since they are similar in structure. The scoring matrix in Figure 2.6 models the considerations above. Note that the table is symmetric - this supports our time-reversible design.

[4]Edit-distance or Levenshtein distance is a metric for measuring the amount of difference between two sequences (e.g., the Levenshtein distance applied to two strings represents the minimum number of edits necessary for transforming one string into another).

[5]RNA polymerase is an enzyme that helps transcribe a nucleotide sequence into mRNA.

	A	G	T	C
A	+1	-½	-1	-1
G	-½	+1	-1	-1
T	-1	-1	+1	-½
C	-1	-1	-½	+1

Figure 2.6: Cost matrix for matches and mismatches

Calculating the scores implies alternating between the probabilistic interpretation of how often biological events occur and the algorithmic interpretation of assigning a score for every operation. The problem is to the find the least expensive (as per the cost matrix) operation sequence which can transform the initial nucleotide sequence into the final nucleotide sequence.

2.3.4 Formulation 4: Varying Gap Cost Models

Biologically, the cost of creating a gap is more expensive than the cost of extending an already created gap[6]. Thus, we could create a model that accounts for this cost variation. There are many such models we could use, including the following:

- **Linear gap penalty:** Fixed cost for all gaps (same as formulation 3).
- **Affine gap penalty:** Impose a large initial cost for opening a gap, then a small incremental cost for each gap extension.
- **General gap penalty:** Allow any cost function. Note this may change the asymptotic runtime of our algorithm.
- **Frame-aware gap penalty:** Tailor the cost function to take into account disruptions to the coding frame.

2.3.5 Enumeration

Recall that in order to solve the Longest Common Substring formulation, we could simply enumerate all possible alignments, evaluate each one, and select the best. This was because there were only $O(n)$ alignments of the two sequences. Once we allow gaps in our alignment, however, this is no longer the case. It is a known issue that the number of all possible gapped alignments cannot be enumerated (at least when the sequences are lengthy). For example, with two sequences of length 1000, the number of possible alignments exceeds the number of atoms in the universe.

Given a metric to score a given alignment, the simple brute-force algorithm enumerates all possible alignments, computes the score of each one, and picks the alignment with the maximum score. This leads to the question, 'How many possible alignments are there?' If you consider only NBAs[7] and $n > m$, the number of alignments is

$$\binom{n}{m} = \frac{(n+m)!}{n!m!} \approx \frac{(2n)!}{(n!)^2} \approx \frac{\sqrt{4\pi n}\frac{(2n)^{2n}}{e^{2n}}}{(\sqrt{2\pi n}\frac{(n)^n}{e^n})} = \frac{2^{2n}}{\sqrt{\pi n}} \tag{2.1}$$

For some small values of n such as 100, the number of alignments is already too big ($> 10^{60}$) for this enumeration strategy to be feasible. Thus, using a better algorithm than brute-force is a necessity.

[6]Insertions and deletions (indels) are rare, therefore it is more likely that a single, longer indel will occur than multiple shorter indels.

[7]Non-Boring Alignments, or alignments where gaps are always paired with nucleotides.

2.4 Dynamic Programming

Before proceeding to a solution of the sequence alignment problem, we first discuss dynamic programming, a general and powerful method for solving problems with certain types of structure.

2.4.1 Theory of Dynamic Programming

Dynamic programming may be used to solve problems with:

1. **Optimal Substructure**: The optimal solution to an instance of the problem contains optimal solutions to subproblems.

2. **Overlapping Subproblems**: There are a limited number of subproblems, many/most of which are repeated many times.

Dynamic programming is usually, but not always, used to solve optimization problems, similar to greedy algorithms. Unlike greedy algorithms, which require a greedy choice property to be valid, dynamic programming works on a range of problems in which locally optimal choices do not produce globally optimal results. Appendix 2.11.3 discusses the distinction between greedy algorithms and dynamic programming in more detail; generally speaking, greedy algorithms solve a smaller class of problems than dynamic programming.

In practice, solving a problem using dynamic programming involves two main parts: Setting up dynamic programming and then performing computation. Setting up dynamic programming usually requires the following 5 steps:

1. Find a 'matrix' parameterization of the problem. Determine the number of dimensions (variables).

2. Ensure the subproblem space is polynomial (not exponential). Note that if a small portion of subproblems are used, then memoization may be better; similarly, if subproblem reuse is not extensive, dynamic programming may not be the best solution for the problem.

3. Determine an effective transversal order. Subproblems must be ready (solved) when they are needed, so computation order matters.

4. Determine a recursive formula: A larger problem is typically solved as a function of its subparts.

5. Remember choices: Typically, the recursive formula involves a minimization or maximization step. Moreover, a representation for storing transversal pointers is often needed, and the representation should be polynomial.

Once dynamic programming is setup, computation is typically straight-forward:

1. Systematically fill in the table of results (and usually traceback pointers) and find an optimal score.

2. Traceback from the optimal score through the pointers to determine an optimal solution.

2.4.2 Fibonacci Numbers

The Fibonacci numbers provide an instructive example of the benefits of dynamic programming. The Fibonacci sequence is recursively defined as $F_0 = F_1 = 1, F_n = F_{n-1} + F_{n-2}$ for $n \leq 2$. We develop an algorithm to compute the n^{th} Fibonacci number, and then refine it first using memoization and later using dynamic programming to illustrate key concepts.

The Naïve Solution

The simple top-down approach is to just apply the recursive definition. Listing 1 shows a simple Python implementation.

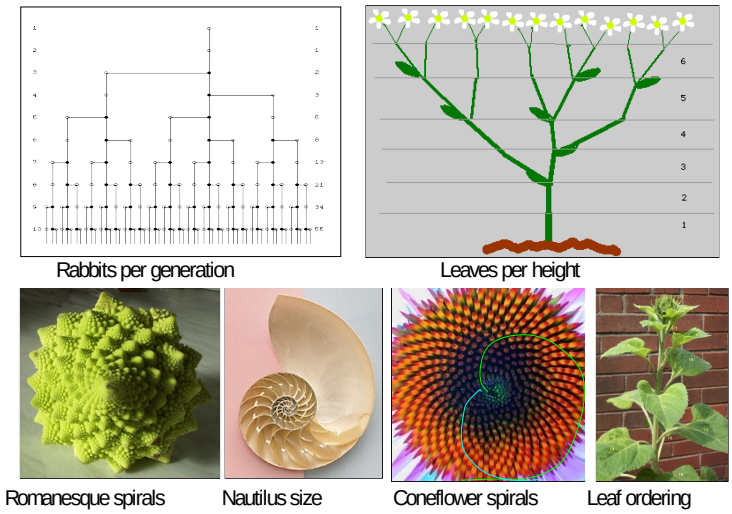


Figure 2.7: Examples of Finonacci numbers in nature are ubiquitous.

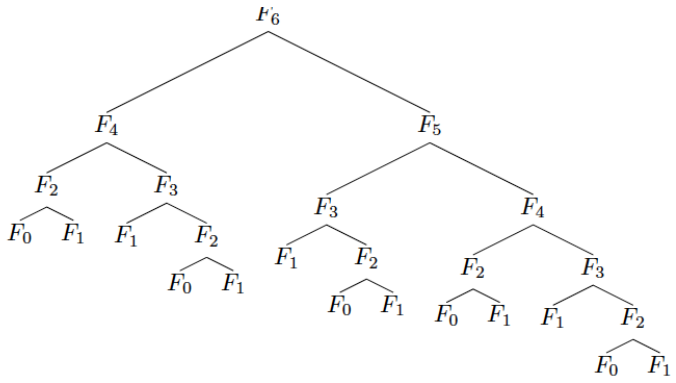


Figure 2.8: The recursion tree for the fib procedure showing repeated subproblems. The size of the tree is $O(\phi(n))$, where ϕ is the golden ratio.

Listing 2.1: Python implementation for computing Fibonacci numbers recursively.

```
# Assume n is a non-negative integer.
def fib(n):
  if n==0 or n==1:
    return 1
  else:
    return fib(n-1) + fib(n-2)
```

But this top-down algorithm runs in exponential time. That is, if $T(n)$ is how long it takes to compute the n^{th} Fibonacci number, we have that $T(n) = T(n-1) + T(n-2)$, so $T(n) = O(\phi^n)$ [8]. The problem is that we are repeating work by solving the same subproblem many times.

[8] ϕ is the **golden ratio**, i.e. $\frac{1+\sqrt{5}}{2}$

The Memoization Solution

A better solution that still utilizes the top-down approach is to memoize the answers to the subproblems. Listing 2 gives a Python implementation that uses memoization.

Listing 2.2: Python implementation for computing Fibonacci numbers using memoization.

```
# Assume n is a non-negative integer.
fibs = {0:1,1:1} # stores subproblem answers
def fib(n):
  if not(n in fibs):
    x = fib(n-2)
    y = fib(n-1)
    fibs[n] = x+y
  return fibs[n]
```

Note that this implementation now runs in $T(n) = O(n)$ time because each subproblem is computed at most once.

The Dynamic Programming Solution

For calculating the n^{th} Fibonacci number, instead of beginning with $F(n)$ and using recursion, we can start computation from the bottom since we know we are going to need all of the subproblems anyway. In this way, we will omit much of the repeated work that would be done by the naïve top-down approach, and we will be able to compute the n^{th} Fibonacci number in $O(n)$ time.

As a formal exercise, we can apply the steps outlined in section 2.4.1:

1. **Find a 'matrix' parameterization:** In this case, the matrix is one-dimensional; there is only one parameter to any subproblem $F(x)$.
2. **Ensure the subproblem space is polynomial:** Since there are only $n-1$ subproblems, the space is polynomial.
3. **Determine an effective transversal order:** As mentioned above, we will apply a bottom-up transversal order (that is, compute the subproblems in order).
4. **Determine a recursive formula:** This is simply the well-known recurrance $F(n) = F(n-1)+F(n-2)$.
5. **Remember choices:** In this case there is nothing to remember, as no choices were made in the recursive formula.

Listing 3 shows a Python implementation of this approach.

Listing 2.3: Python implementation for computing Fibonacci numbers iteratively using dynamic programming.

```
# Assume n is a non-negative integer
def fib(n):
  x = y = 1
  for i in range(1,n):
    x, y = y, x + y
  return x
```

This method is optimized to only use constant space instead of an entire table since we only need the answer to each subproblem once. But in general dynamic programming solutions, we want to store the solutions to subproblems in a table since we may need to use them multiple times without recomputing their answers. Such solutions would look somewhat like the memoization solution in Listing 2, but they will generally be bottom-up instead of top-down. In this particular example, the distinction between the memoization solution and the dynamic programming solution is minimal as both approaches compute all subproblem solutions and use them the same number of times. In general, memoization is useful when not all

subproblems will be computed, while dynamic programming may not have as much overhead as memoization when all subproblem solutions must be calculated. Additional dynamic programming examples may be found online [6].

2.4.3 Sequence Alignment using Dynamic Programming

We are now ready to solve the more difficult problem of sequence alignment using dynamic programming, which is presented in depth in the next section. Note that the key insight in solving the sequence alignment problem is that alignment scores are additive. This allows us to create a matrix M indexed by i and j, which are positions in two sequences S and T to be aligned. The best alignment of S and T corresponds with the best path through the matrix M after it is filled in using a recursive formula.

By using dynamic programming to solve the sequence alignment problem, we achieve a provably optimal solution that is far more tractable than brute-force enumeration.

2.5 The Needleman-Wunsch Algorithm

We will now use dynamic programming to tackle the harder problem of general sequence alignment. Given two strings S $=(S_1, \ldots, S_n)$ and T $=(T_1, \ldots, T_m)$, we want to find the longest common subsequence, which may or may not contain gaps. Rather than maximizing the length of a common subsequence we want to compute the common subsequence that optimizes the score as defined by our scoring function. Let d denote the gap penalty cost and s(x; y) the score of aligning a base x and a base y. These are inferred from insertion/deletion and substitution probabilities which can be determined experimentally or by looking at sequences that we know are closely related. The algorithm we will develop in the following sections to solve sequence alignment is known as the Needleman-Wunsch algorithm.

2.5.1 Dynamic programming vs. memoization

Before we dive into the algorithm, a final note on memoization[9] is in order. Much like the Fibonacci problem, the sequence alignment problem can be solved in either a top-down or bottom-up approach.

In a *top-down recursive approach* we can use memoization to create a potentially large dictionary indexed by each of the subproblems that we are solving (aligned sequences). This needs $O(n^2m^2)$ space if we index each subproblem by the starting and end points of the subsequences for which an optimal alignment needs to be computed. The advantage is that we solve each subproblem at most once: if it is not in the dictionary, the problem gets computed and then inserted into dictionary for further reference.

In a *bottom-up iterative approach* we can use dynamic programming. We define the order of computing sub-problems in such a way that a solution to a problem is computed once the relevant sub-problems have been solved. In particular, simpler sub-problems will come before more complex ones. This removes the need for keeping track of which sub-problems have been solved (the dictionary in memoization turns into a matrix) and ensures that there is no duplicated work (each sub-alignment is computed only once).

2.5.2 Problem Statement

Suppose we have an optimal alignment for two sequences S and T in which S_i matches T_j. The key insight is that this optimal alignment is composed of an optimal alignment between $(S_1, \ldots, S_{i-1})$ and $(T_1, \ldots, T_{j-1})$ and an optimal alignment between $(S_{i+1}, \ldots, S_n)$ and $(T_{j+1}, \ldots, T_m)$. This follows from a cut-and-paste argument: if one of these partial alignments is suboptimal, then we cut-and-paste a better alignment in place of the suboptimal one. This achieves a higher score of the overall alignment and thus contradicts the optimality of the initial global alignment. In other words, every subpath in an optimal path must also be optimal. Notice that the scores are additive, so the score of the overall alignment equals the addition of the scores of the alignments of the subsequences. This implicitly assumes that the sub-problems of computing the optimal scoring alignments of the subsequences are independent. We need to biologically motivate that such an assumption leads to meaningful results.

[9] Memoization is a technique that reduces processing time by storing the answers to calculations or subproblems that are called many times.

2.5.3 Index space of subproblems

We now need to index the space of subproblems. Let $F_{i,j}$ be the score of the optimal alignment of $(S_1, \ldots, S_i)$ and $(T_1, \ldots, T_j)$. The space of subproblems is $\{F_{i,j}\ \forall i \in [1, |S|], \forall j \in [1, |T|]\}$. This allows us to maintain an $m \times n$ matrix F with the solutions (i.e. optimal scores) for all the subproblems.

2.5.4 Local optimality

We can compute the optimal solution for a subproblem by making a locally optimal choice based on the results from the smaller sub-problems. Thus, we need to establish a recursive function that shows how the solution to a given problem depends on its subproblems. And we use this recursive definition to fill up the table F in a bottom-up fashion.

We can consider the 4 possibilities (insert, delete, substitute, match) and evaluate each of them based on the results we have computed for smaller subproblems. To initialize the table, we set $F_{0,j} = -j \cdot d$ $F_{i,0} = -i \cdot d$ since those are the scores of aligning $(T_1, \ldots, T_j\ )$ with j gaps and $(S_1, \ldots., S_i)$ with i gaps (aka zero overlap between the two sequences). Then we traverse the matrix column by column computing the optimal score for each alignment subproblem by considering the four possibilities:

- Sequence S has a gap at the current alignment position.
- Sequence T has a gap at the current alignment position.
- There is a mutation (nucleotide substitution) at the current position.
- There is a match at the current position.

We then use the possibility that produces the maximum score. We express this mathematically by the recursive formula for $F_{i,j}$:

$$\begin{aligned} \textit{Initialization} \quad &: \quad \begin{array}{l} F(0,0) = 0 \\ F(i,0) = F(i-1,0) - d \\ F(0,j) = F(0,j-1) - d \end{array} \\ \textit{Iteration} \quad &: \quad F(i,j) = \max \begin{cases} F(i-1,j) - d & \text{insert gap in } S \\ F(i,j-1) - d & \text{insert gap in } T \\ F(i-1,j-1) + s(x_i, y_j) & \text{match or mutation} \end{cases} \\ \textit{Termination} \quad &: \quad \textit{Bottom right} \end{aligned}$$

After traversing the matrix, the optimal score for the global alignment is given by $F_{m,n}$. The traversal order needs to be such that we have solutions to given subproblems when we need them. Namely, to compute $F_{i,j}$, we need to know the values to the left, up, and diagonally above $F_{i,j}$ in the table. Thus we can traverse the table in row or column major order or even diagonally from the top left cell to the bottom right cell. Now, to obtain the actual alignment we just have to remember the choices that we made at each step.

2.5.5 Optimal Solution

Note the duality between paths through the matrix F and sequence alignments. In evaluating each cell $F_{i,j}$ we make a choice by selecting the maximum of the three possibilities. Thus the value of each (uninitialized) cell in the matrix is determined either by the cell to its left, above it, or diagonally to the left above it. A match and a substitution are both represented as traveling in the diagonal direction; however, a different cost can be applied for each, depending on whether the two base pairs we are aligning match or not. To construct the actual optimal alignment, we need to traceback through our choices in the matrix. It is helpful to maintain a pointer for each cell while filling up the table that shows which choice was made to get the score for that cell. Then we can just follow our pointers backwards to reconstruct the optimal alignment.

2.5.6 Solution Analysis

The runtime analysis of this algorithm is very simple. Each update takes $O(1)$ time, and since there are mn elements in the matrix F, the total running time is $O(mn)$. Similarly, the total storage space is $O(mn)$. For the more general case where the update rule is more complicated, the running time may be more expensive. For instance, if the update rule requires testing all sizes of gaps (e.g. the cost of a gap is not linear), then the running time would be $O(mn(m+n))$.

2.5.7 Needleman-Wunsch in practice

Assume we want to align two sequences S and T, where

S = AGT

T = AAGC

The first step is placing the two sequences along the margins of a matrix and initializing the matrix cells. To initialize we assign a 0 to the first entry in the matrix and then fill in the first row and column based on the incremental addition of gap penalties, as in Figure 2.9 below. Although the algorithm could fill in the first row and column through iteration, it is important to clearly define and set boundaries on the problem.

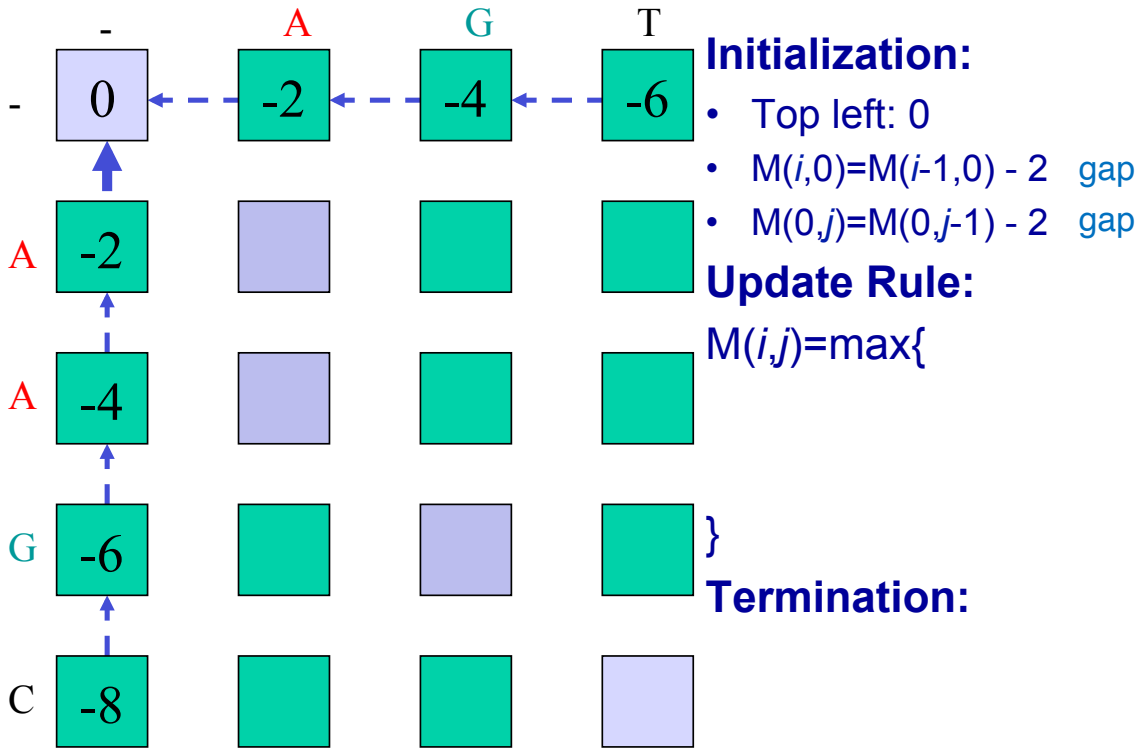


Figure 2.9: (Example) Initial setup for Needleman-Wunsch

The next step is iteration through the matrix. The algorithm proceeds either along rows or along columns, considering one cell at time. For each cell three scores are calculated, depending on the scores of three adjacent matrix cells (specifically the entry above, the one diagonally up and to the left, and the one to the left). The maximum score of these three possible tracebacks is assigned to the entry and the corresponding pointer is also stored. Termination occurs when the algorithm reaches the bottom right corner. In Figure 2.10 the alignment matrix for sequences S and T has been filled in with scores and pointers.

The final step of the algorithm is optimal path traceback. In our example we start at the bottom right corner and follow the available pointers to the top left corner. By recording the alignment decisions made at each cell during traceback, we can reconstruct the optimal sequence alignment from end to beginning and

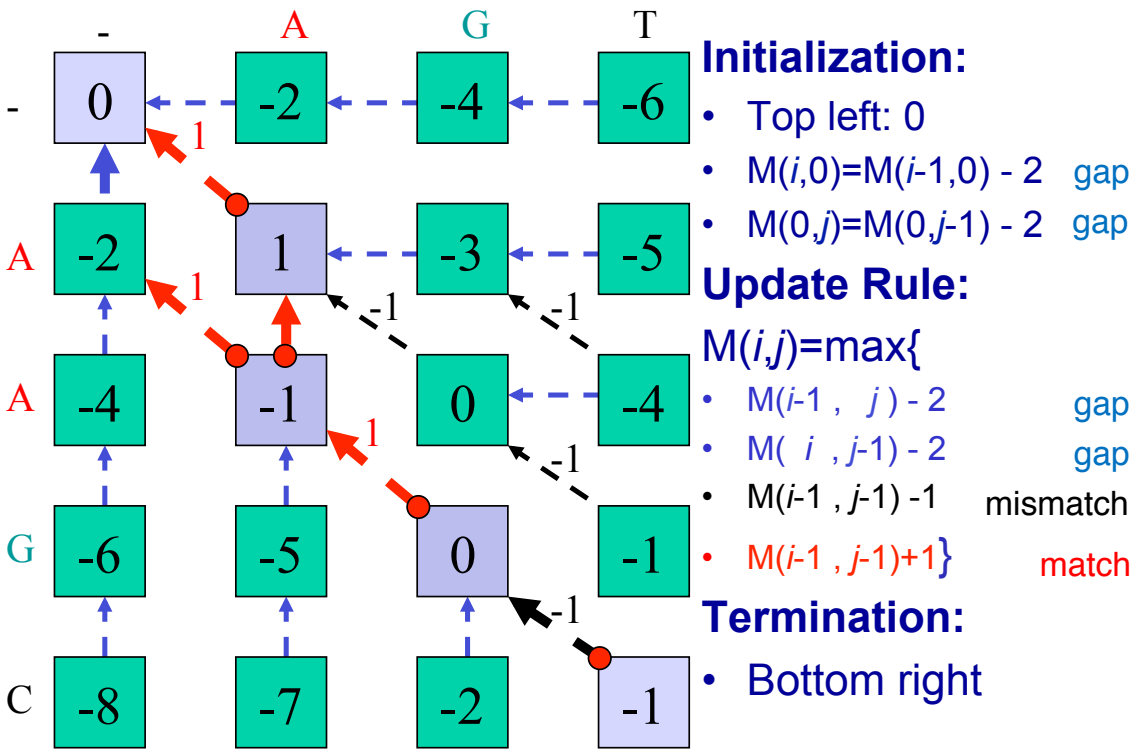


Figure 2.10: (Example) Half-way through the second step of Needleman-Wunsch

then invert it. Note that in this particular case, multiple optimal pathways are possible (Figure 2.11). A pseudocode implementation of the Needleman-Wunsch algorithm is included in Appendix 2.11.4

2.5.8 Optimizations

The algorithm we presented is much faster than the brute-force strategy of enumerating alignments and it performs well for sequences up to 10 kilo-bases long. Nevertheless, at the scale of whole genome alignments the algorithm given is not feasible. In order to align much larger sequences we can make modifications to the algorithm and further improve its performance.

Bounded Dynamic Programming

One possible optimization is to ignore Mildly Boring Alignments (MBAs), or alignments that have too many gaps. Explicitly, we can limit ourselves to stay within some distance W from the diagonal in the matrix F of subproblems. That is, we assume that the optimizing path in F from $F_{0,0}$ to $F_{m,n}$ is within distance W along the diagonal. This means that recursion (2.2) only needs to be applied to the entries in F within distance W around the diagonal, and this yields a time/space cost of $O((m+n)W)$ (refer to Figure 2.12).

Note, however, that this strategy is heuristic and no longer guarantees an optimal alignment. Instead it attains a lower bound on the optimal score. This can be used in a subsequent step where we discard the recursions in matrix F which, given the lower bound, cannot lead to an optimal alignment.

Linear Space Alignment

Recursion (2.2) can be solved using only linear space: we update the columns in F from left to right during which we only keep track of the last updated column which costs $O(m)$ space. However, besides the score $F_{m,n}$ of the optimal alignment, we also want to compute a corresponding alignment. If we use trace back, then we need to store pointers for each of the entries in F, and this costs $O(mn)$ space.

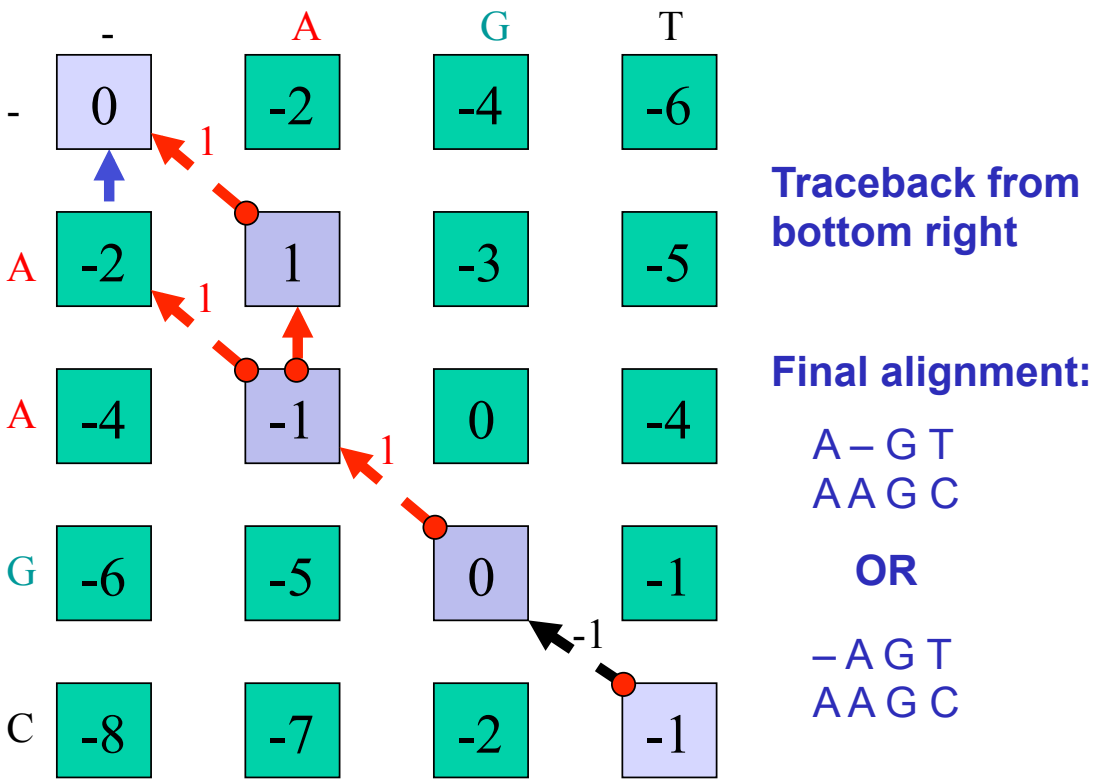


Figure 2.11: (Example) Tracing the optimal alignment

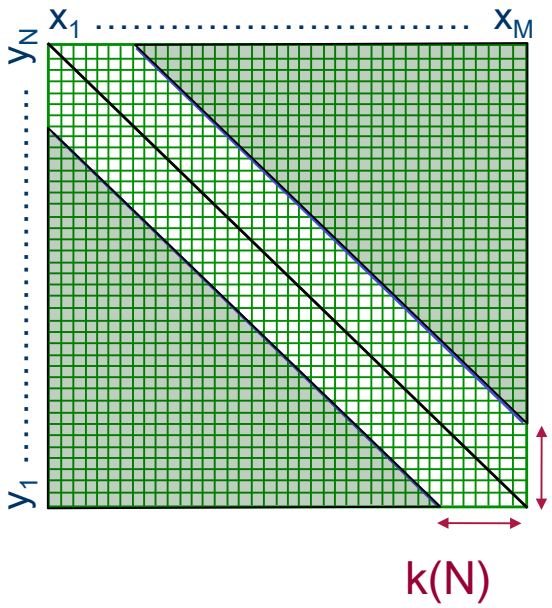


Figure 2.12: Bounded dynamic programming example

It is also possible to find an optimal alignment using only linear space! The goal is to use divide and conquer in order to compute the structure of the optimal alignment for one matrix entry in each step. Figure 2.13 illustrates the process. The key idea is that a dynamic programming alignment can proceed just as

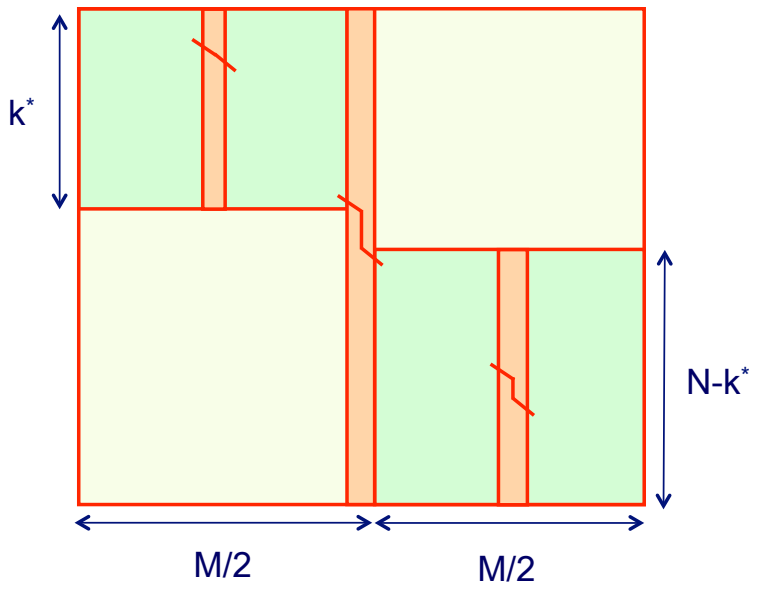


Figure 2.13: Recovering the sequence alignment with $O(m+n)$ space

easily in the reverse direction, starting at the bottom right corner and terminating at the top left. So if the matrix is divided in half, then both a forward pass and a reverse pass can run at the same time and converge in the middle column. At the crossing point we can add the two alignment scores together; the cell in the middle column with the maximum score must fall in the overall optimal path.

We can describe this process more formally and quantitatively. First compute the row index $u \in \{1, \ldots, m\}$ that is on the optimal path while crossing the $\frac{n}{2}^{\text{th}}$ column. For $1 \leq i \leq m$ and $\frac{n}{2} \leq j \leq n$ let $C_{i,j}$ denote the row index that is on the optimal path to $F_{i,j}$ while crossing the $\frac{n}{2}^{\text{th}}$ column. Then, while we update the columns of F from left to right, we can also update the columns of C from left to right. So, in $O(mn)$ time and $O(m)$ space we are able to compute the score $F_{m,n}$ and also $C_{m,n}$, which is equal to the row index $u \in \{1, \ldots, m\}$ that is on the optimal path while crossing the $\frac{n}{2}^{\text{th}}$ column.

Now the idea of divide and conquer kicks in. We repeat the above procedure for the upper left $u \times \frac{n}{2}$ submatrix of F and also repeat the above procedure for the lower right $(m-u) \times \frac{n}{2}$ submatrix of F . This can be done using $O(m+n)$ allocated linear space. The running time for the upper left submatrix is $O(\frac{un}{2})$ and the running time for the lower right submatrix is $O(\frac{(m-u)n}{2})$, which added together gives a running time of $O(\frac{mn}{2}) = O(mn)$.

We keep on repeating the above procedure for smaller and smaller submatrices of F while we gather more and more entries of an alignment with optimal score. The total running time is $O(mn)+O(\frac{mn}{2})+O(\frac{mn}{4})+\ldots = O(2mn) = O(mn)$. So, without sacrificing the overall running time (up to a constant factor), divide and conquer leads to a linear space solution.

2.6 Multiple alignment

2.6.1 Aligning three sequences

Now that we have seen how to align a pair of sequences, it is natural to extend this idea to *multiple* sequences. Suppose we would like to find the optimal alignment of 3 sequences. How might we proceed?

Recall that when we align two sequences S and T, we choose the maximum of three possibilities for the final position of the alignment (sequence T aligned against a gap, sequence S aligned against a gap, or

sequence S aligned against sequence T):

$$F_{i,j} = \max \begin{cases} F_{i,j-1} + d \\ F_{i-1,j} + d \\ F_{i-1,j-1} + s(S_i, T_j) \end{cases}$$

For three sequences S,T, and U, there are seven possibilities for the final position of the alignment. That is, there are three ways to have two gaps in the final position, three ways to have one gap, and one way to have all three sequences aligned ($\binom{3}{1} + \binom{3}{2} + \binom{3}{3} = 7$). The update rule is now:

$$F_{i,j,k} = \max \begin{cases} F_{i-1,j,k} + s(S_i, -, -) \\ F_{i,j-1,k} + s(-, T_j, -) \\ F_{i,j,k-1} + s(-, -, U_k) \\ F_{i-1,j-1,k} + s(S_i, T_j, -) \\ F_{i-1,j,k-1} + s(S_i, -, U_k) \\ F_{i,j-1,k-1} + s(-, T_j, U_k) \\ F_{i-1,j-1,k-1} + s(S_i, T_j, U_k) \end{cases}$$

where s is the function describing gap, match, and mismatch scores.

This approach, however, is exponential in the number of sequences we are aligning. If we have k sequences of length n, computing the optimal alignment using a k-dimensional dynamic programming matrix takes $O((2n)^k)$ time (the factor of 2 results from the fact that a k-cube has 2^k vertices, so we need to take the maximum of $2^k - 1$ neighboring cells for each entry in the score matrix). As you can imagine, this algorithm quickly becomes impractical as the number of sequences increases.

2.6.2 Heuristic multiple alignment

One commonly used approach for multiple sequence alignment is called *progressive multiple alignment*. Assume that we know the evolutionary tree relating each of our sequences. Then we begin by performing a pairwise alignment of the two most closely-related sequences. This initial alignment is called the *seed alignment*. We then proceed to align the next closest sequence to the seed, and this new alignment replaces the seed. This process continues until the final alignment is produced.

In practice, we generally do not know the evolutionary tree (or *guide tree*), this technique is usually paired with some sort of clustering algorithm that may use a low-resolution similarity measure to generate an estimation of the tree.

While the running time of this heuristic approach is much improved over the previous method (polynomial in the number of sequences rather than exponential), we can no longer guarantee that the final alignment is optimal.

Note that we have not yet explained how to align a sequence against an existing alignment. One possible approach would be to perform pairwise alignments of the new sequence with each sequence already in the seed alignment (we assume that any position in the seed alignment that is already a gap will remain one). Then we can add the new sequence onto the seed alignment based on the best pairwise alignment (this approach was previously described by Feng and Doolittle[4]). Alternatively, we can devise a function for scoring the alignment of a sequence with another alignment (such scoring functions are often based on the pairwise sum of the scores at each position).

Design of better multiple sequence alignment tools is an active area of research. Section 2.9 details some of the current work in this field.

2.7 Current Research Directions

2.8 Further Reading

2.9 Tools and Techniques

Lalign finds local alignments between two sequences. Dotlet is a browser-based Java applet for visualizing the alignment of two sequences in a dot-matrix.

The following tools are available for multiple sequence alignment:

- Clustal Omega - A multiple sequence alignment program that uses seeded guide trees and HMM profile-profile techniques to generate alignments.[9]
- MUSCLE - MUltiple Sequence Comparison by Log-Expectation[3]
- T-Coffee - Allows you to combine results obtained with several alignment methods[2]
- MAFFT - (Multiple Alignment using Fast Fourier Transform) is a high speed multiple sequence alignment program[5]
- Kalign - A fast and accurate multiple sequence alignment algorithm[8]

2.10 What Have We Learned?

2.11 Appendix

2.11.1 Homology

One of the key goals of sequence alignment is to identify homologous sequences (e.g., genes) in a genome. Two sequences that are homologous are evolutionarily related, specifically by descent from a common ancestor. The two primary types of homologs are orthologous and paralogous (refer to Figure 2.14[10]). Other forms of homology exist (e.g., xenologs), but they are outside the scope of these notes.

Orthologs arise from speciation events, leading to two organisms with a copy of the same gene. For example, when a single species A speciates into two species B and C, there are genes in species B and C that descend from a common gene in species A, and these genes in B and C are orthologous (the genes continue to evolve independent of each other, but still perform the same relative function).

Paralogs arise from duplication events within a species. For example, when a gene duplication occurs in some species A, the species has an original gene B and a gene copy B′, and the genes B and B′ are paralogus.

Generally, orthologous sequences between two species will be more closely related to each other than paralogous sequences. This occurs because orthologous typically (although not always) preserve function over time, whereas paralogous often change over time, for example by specializing a gene's (sub)function or by evolving a new function. As a result, determining orthologous sequences is generally more important than identifying paralogous sequences when gauging evolutionary relatedness.

2.11.2 Natural Selection

The topic of natural selection is a too large topic to summarize effectively in just a few short paragraphs; instead, this appendix introduces three broad types of natural selection: positive selection, negative selection, and neutral selection.

- *Positive selection* occurs when a trait is evolutionarily advantageous and increases an individual's fitness, so that an individual with the trait is more likely to have (robust) offspring. It is often associated with the development of new traits.

[10] R.B. - BIOS 60579

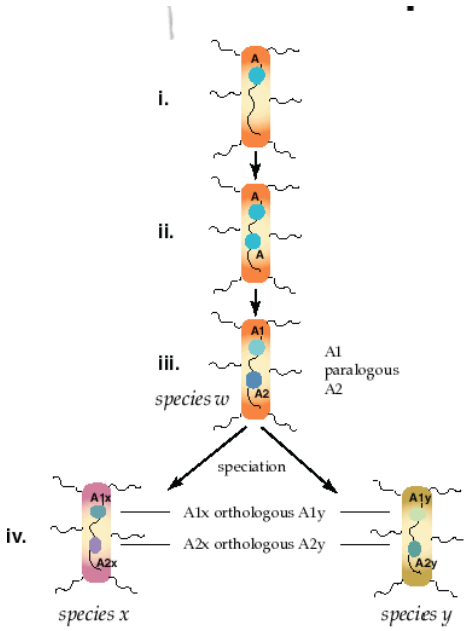


Figure 2.14: Ortholog and paralog sequences

- *Negative selection* occurs when a trait is evolutionarily disadvantageous and decreases an individual's fitness. Negative selection acts to reduce the prevalence of genetic alleles that reduce a species' fitness. Negative selection is also known as purifying selection due to its tendency to 'purify' genetic alleles until only a single allele exists in the population.
- *Neutral selection* describes evolution that occurs randomly, as a result of alleles not affecting an individual's fitness. In the absence of selective pressures, no positive or negative selection occurs, and the result is neutral selection.

2.11.3 Dynamic Programming v. Greedy Algorithms

Dynamic programming and greedy algorithms are somewhat similar, and it behooves one to know the distinctions between the two. Problems that may be solved using dynamic programming are typically optimization problems that exhibit two traits:

1. **optimal substructure** and
2. **overlapping subproblems**.

Problems solvable by greedy algorithms require both these traits as well as (3) the **greedy choice property**. When dealing with a problem "in the wild," it is often easy to determine whether it satisfies (1) and (2) but difficult to determine whether it must have the greedy choice property. It is not always clear whether locally optimal choices will yield a globally optimal solution.

For computational biologists, there are two useful points to note concerning whether to employ dynamic programming or greedy programming. First, if a problem may be solved using a greedy algorithm, then it may be solved using dynamic programming, while the converse is not true. Second, the problem structures that allow for greedy algorithms typically do not appear in computational biology.

To elucidate this second point, it could be useful to consider the structures that allow greedy programming to work, but such a discussion would take us too far afield. The interested student (preferably one with a mathematical background) should look at matroids and greedoids, which are structures that have the greedy choice property. For our purposes, we will simply state that biological problems typically involve entities that are highly systemic and that there is little reason to suspect sufficient structure in most problems to employ greedy algorithms.

2.11.4 Pseudocode for the Needleman-Wunsch Algorithm

The first problem in the first problem set asks you to finish an implementation of the Needleman-Wunsch (NW) algorithm, and working Python code for the algorithm is intentionally omitted. Instead, this appendix summarizes the general steps of the NW algorithm (Section 2.5) in a single place.

Problem: Given two sequences S and T of length m and n, a substitution matrix vU of matching scores, and a gap penalty G, determine the optimal alignment of S and T and the score of the alignment.

Algorithm:

1. Create two $m+1$ by $n+1$ matrices A and B. A will be the scoring matrix, and B will be the traceback matrix. The entry (i, j) of matrix A will hold the score of the optimal alignment of the sequences $S[1, \ldots, i]$ and $T[1, \ldots, j]$, and the entry (i, j) of matrix B will hold a pointer to the entry from which the optimal alignment was built.

2. Initialize the first row and column of the score matrix A such that the scores account for gap penalties, and initialize the first row and column of the traceback matrix B in the obvious way.

3. Go through the entries (i, j) of matrix A in some reasonable order, determining the optimal alignment of the sequences $S[1, \ldots, i]$ and $T[1, \ldots, j]$ using the entries $(i-1, j-1)$, $(i-1, j)$, and $(i, j-1)$. Set the pointer in the matrix B to the corresponding entry from which the optimal alignment at (i, j) was built.

4. Once all entries of matrices A and B are completed, the score of the optimal alignment may be found in entry (m, n) of matrix A.

5. Construct the optimal alignment by following the path of pointers starting at entry (m, n) of matrix B and ending at entry $(0, 0)$ of matrix B.

Bibliography

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, London, third edition, 1964.

[2] Paolo Di Tommaso, Sebastien Moretti, Ioannis Xenarios, Miquel Orobitg, Alberto Montanyola, Jia-Ming Chang, Jean-François Taly, and Cedric Notredame. T-Coffee: a web server for the multiple sequence alignment of protein and RNA sequences using structural information and homology extension. *Nucleic Acids Research*, 39(Web Server issue):W13–W17, 2011.

[3] Robert C Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–7, January 2004.

[4] D F Feng and R F Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, 1987.

[5] Kazutaka Katoh, George Asimenos, and Hiroyuki Toh. Multiple alignment of DNA sequences with MAFFT. *Methods In Molecular Biology Clifton Nj*, 537:39–64, 2009.

[6] Manolis Kellis. Dynamic programming practice problems. http://people.csail.mit.edu/bdean/6.046/dp/, September 2010.

[7] Manolis Kellis, Nick Patterson, Matthew Endrizzi, Bruce Birren, and Eric S Lander. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423(6937):241–254, 2003.

[8] Timo Lassmann and Erik L L Sonnhammer. Kalign–an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformatics*, 6(1):298, 2005.

[9] Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, Julie D Thompson, and Desmond G Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(539):539, 2011.

CHAPTER

THREE

RAPID SEQUENCE ALIGNMENT AND DATABASE SEARCH

Maria Rodriguez (Sep 13, 2011)
Rushil Goel (Sep 16, 2010)
Eric Eisner -Guilhelm Richard (Sep 17, 2009)
Tural Badirkhnali (Sep 11, 2008)

Figures

3.1 Introduction

In the previous chapter, we used dynamic programming to compute sequence alignments in $O(n^2)$. In particular, you learned the Needleman-Wunsch algorithm for global alignment, which matches completes sequence with one another at the nucleotide level when they are known to be homologous (i.e. the sequences come from organisms that share a common ancestor).

The biological significance of finding sequence alignments is to be able to infer the most likely set of evolutionary events such as point mutations/mismatches and gaps (insertions or deletions) that occured in order to transform one sequence into the other. By assigning costs to each transformation type (mismatch or gap) that reflect their respective levels of evolutionary difficulty, finding an optimal alignment reduces to finding the set of transformations that result in the lowest overall cost.

A dynamic algorithm, which uses optimal substructure to decompose a problem into similar sub-problems. The problem of finding a sequence alignment can be nicely expressed as a dynamic programming algorithm since alignment scores are additive, which means that finding the alignment of a larger sequence can be found by recursively finding the alignment to a smaller sequence. The scores are stored in a matrix, with

one sequence corresponding to the columns and the other sequence corresponding to the rows. Each cell represents the transformation required between two nucleotides corresponding to the cell's row and column. In contrast to a dynamic programming algorithm, a greedy algorithm simply chooses the transition with minimum cost at each step, which does not guarantee that the overall result will give the optimal or lowest-cost alignment. An alignment is recovered by tracing back through the dynamic programming matrix (shown below) in the global alignment algorithm.

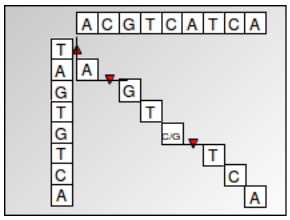


Figure 3.1: Global Alignment

To summarize the Needleman-Wunsch algorithm for global alignment:

While computing scores corresponding to each cell in the matrix, we remember our choice (memoization) at that step i.e. which one of the top, left or diagonal cells led to the maximum score for the current cell. So, at the end, we are left with a matrix full of optimal scores at each cell position, along with pointers at each cell reflecting the optimal choice that leads to that particular cell.

The last step involves a traceback from the cell in the bottom right corner (which contains the score of aligning one complete sequence with the other) by following the pointers reflecting locally optimal choices, and then constructing the alignment corresponding to an optimal path followed in the matrix.

The running time of Needleman-Wunsch algorithm $O(n^2)$ since for each cell in the matrix, we do a finite amount of computation i.e. we calculate 3 values using already computed scores and then take the maximum of those values to find the score corresponding to that cell.

To guarantee correctness, it is necessary to compute the cost for every cell of the matrix. Strictly speaking, it is possible that the optimal alignment may be made up of a bad alignment (consisting of gaps and mismatches) at the start, followed by a really good alignment that makes it the best alignment overall. Since we do not know where the best alignment will come from, in theory, we need to compute every entry of the matrix to find a provable optimal global alignment. However, in practice, we can often restrict the alignment space to be explored, if we know that some alignments are clearly sub-optimal (this will be discussed in the current lecture). This is a heuristic, and while it may not work all the time since it is not a guaranteed optimal solution, it works remarkably well in practice. It was noted that depending on the properties of the scoring matrix, it may be possible to argue the correctness of the bounded-space algorithm, even though such an algorithm will not be guaranteed to be faster all the time.

In this chapter, we will discuss database search for aligning a newly-sequenced gene to reference genes in a known genome. This chapter will also introduce the Smith-Waterman algorithm for local alignment for aligning subsequences as opposed to complete sequences, in contrast to the Needleman-Wunsch algorithm for global alignment. Later on in the chapter you will be given an overview of Hashing and semi-numerical methods like the Karp-Rabin algorithm for finding the longest (contiguous) common substring of nucleotides. These algorithms are implemented and extended for inexact matching in the BLAST program , one of the most famous and successful tools in computational biology. Finally, this chapter will go over BLAST as well as the probabilistic foundation of sequence alignment and how alignment scores can be interpreted as likelihood ratios.

Outline:

1. **Global alignment vs. Local alignment**

- Variations on initialization, termination, update rule for Global Alignment (Needleman-Wunsch) vs. Local Alignment (Smith-Waterman)
- Varying gap penalties, algorithmic speedups

2. **Linear-time exact string matching**
   - Karp-Rabin algorithm and semi-numerical methods
   - Hash functions and randomized algorithms
3. **The BLAST algorithm and inexact matching**
   - Hashing with neighborhood search
   - Two-hit blast and hashing with combs
4. **Pre-processing for linear-time string matching**
   - Fundamental pre-processing
   - Suffix Trees
   - Suffix Arrays
   - The Burrows-Wheeler Transform
5. **Probabilistic foundations of sequence alignment**
   - Mismatch penalties, BLOSUM and PAM matrices
   - Statistical significance of an alignment score

3.2 Global alignment vs. Local alignment

A **global alignment** is defined as the *end-to-end* alignment of two sequences.

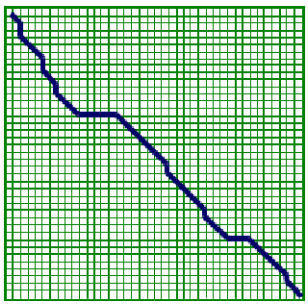


Figure 3.2: Global Alignment

A **local alignment** of string s and t is an alignment of a *substring* of s with a substring of t.

In general, local alignments are used to find regions of high local similarity. Often, we are more interested in finding local alignments because we normally do not know the boundaries of genes and only a small domain of the gene may be conserved. In such a cases, we do not want to enforce that other (potentially non-homologous) parts of the sequence also align. Local alignment is also useful when searching for a small gene in a large chromosome. A local alignment is also useful in such a case where a long sequence may have undergone rearrangements and it may have been broken up into different smaller segments, since a local alignment could detect such rearrangements.

Another type of alignment is **semi-global alignment**.

This form of alignment is useful for overlap detection when we do not wish to penalize starting or ending gaps.

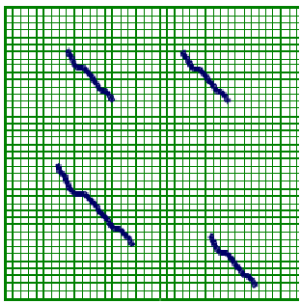


Figure 3.3: Local Alignment

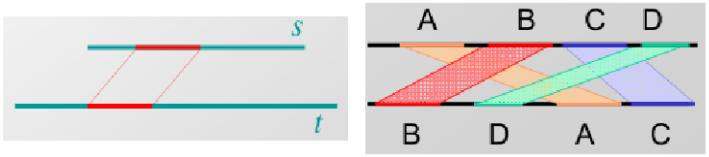


Figure 3.4: Local alignments to detect rearrangements

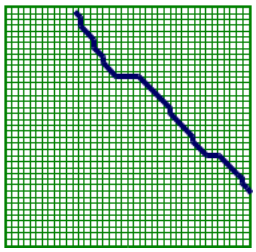


Figure 3.5: Semi-global Alignmnent

3.2.1 Using Dynamic Programming for local alignments

In this section we will see how to find local alignments with a minor modification of Needleman-Wunsch algorithm that was discussed in the previous chapter for finding global alignments.

To find global alignments, we use the following dynamic programming algorithm (Needleman-Wunsch algorithm):

$$
\begin{aligned}
\textit{Initialization} \quad &: \quad F(0,0) = 0 \\
\textit{Iteration} \quad &: \quad F(i,j) = \max \begin{cases} F(i-1,j) - d \\ F(i,j-1) - d \\ F(i-1,j-1) + s(x_i, y_j) \end{cases} \\
\textit{Termination} \quad &: \quad \textit{Bottom right}
\end{aligned}
$$

For finding local alignments we only need to modify the Needleman-Wunsch algorithm slightly to prevent an alignment score from going negative by checking and starting over to find a new local alignment. Since a local alignment can start anywhere, we initialize the first row and column in the matrix to zeros. Furthermore, since the alignment can end anywhere, we need to traverse the entire matrix to find the optimal alignment score (not only in the bottom right corner). The rest of the algorithm, including traceback, remains the unchanged.

To find local alignments we use the following dynamic programming algorithm for local alignment (Smith-Waterman algorithm):

$$
\begin{aligned}
\textit{Initialization} \quad &: \quad F(0,0) = 0 \\
&\qquad\quad F(0,j) = 0 \\
\textit{Iteration} \quad &: \quad F(i,j) = \max \begin{cases} 0 \\ F(i-1,j) - d \\ F(i,j-1) - d \\ F(i-1,j-1) + s(x_i, y_j) \end{cases} \\
\textit{Termination} \quad &: \quad \textit{Anywhere}
\end{aligned}
$$

For finding a semi-global alignment, the important distinctions are to initialize the top row and leftmost column to zero and terminate end at either the bottom row or rightmost column.

To perform a semi-global alignment, the algorithm is as follows:

$$
\begin{aligned}
\textit{Initialization} \quad &: \quad F(i,0) = 0 \\
&\qquad\quad F(0,j) = 0 \\
\textit{Iteration} \quad &: \quad F(i,j) = \max \begin{cases} F(i-1,j) - d \\ F(i,j-1) - d \\ F(i-1,j-1) + s(x_i, y_j) \end{cases} \\
\textit{Termination} \quad &: \quad \textit{Bottom Row or Right Column}
\end{aligned}
$$

3.2.2 Algorithmic Variations

Sometimes it can be costly in both time and space to run these alignment algorithms. Therefore, this section presents some algorithmic variations to save time and space that work well in practice.

One method to save time, which was brought up earlier, is the idea of limiting the space of alignments to be explored. Good alignments generally stay close to the diagonal of the matrix. Thus we can just explore a width of k around the diagonal. As mentioned at the beginning of this chapter, the problem with this modification is that this is a heuristic and can lead to a sub-optimal solution. Nevertheless, this works very well in practice. This algorithm requires $O(k * m)$ space and $O(k * m)$ time.

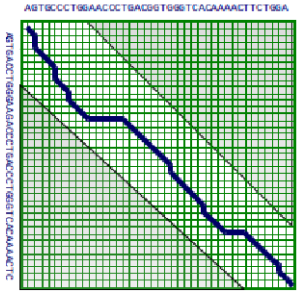


Figure 3.6: Bounded-space computation

We saw earlier that in order to compute the optimal solution, we needed to store the score in each cell, as well as the pointer reflecting the optimal choice leading to each cell. However, if we are only interested in the *optimal alignment score*, and not the actual alignment itself, there is a method to compute the solution while saving space. To compute the score of any cell we only need the scores of the cell above, to the left, and to the left-diagonal of the current cell. By saving the previous and current column in which we are computing scores, the optimal solution can be computed in linear space.

To compute the *optimal alignment*, however, we need quadratic time and linear space. The idea, based on the principle of divide-and-conquer, is that we compute the optimal alignments from both sides of the matrix i.e. from the left to the right, and vice versa. Let $u = \lfloor \frac{n}{2} \rfloor$. Say we can identify v such that cell

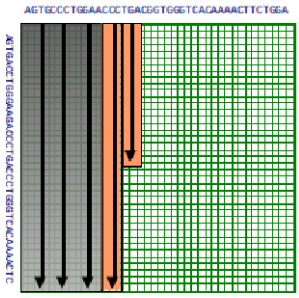


Figure 3.7: Linear-space computation

(u, v) is on the optimal alignment path. That means v is the row where the alignment crosses column u of the matrix. We can find the optimal alignment by concatenating the optimal alignments from (0,0) to (u, v) plus that of (u, v) to (m, n), where m and n is the bottom right cell (note: alignment scores of concatenated subalignments using our scoring scheme are additive. So we have isolated our problem to two separate problems in the the top left and bottom right corners of the DP matrix. Then we can recursively keep dividing up these subproblems to smaller subproblems, until we are down to aligning 0-length sequences or our problem is small enough to apply the regular DP algorithm. To find v the row in the middle column where the optimal alignment crosses we simply add the incoming and outgoing scores for that column.

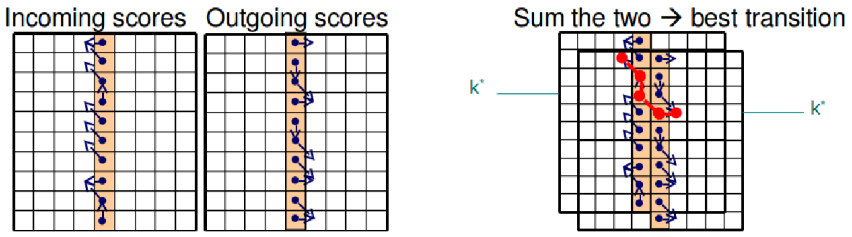


Figure 3.8: Space-saving optimization

One drawback of this divide-and-conquer approach is that it has a longer run-time. Nevertheless, the run-time is not dramatically increased. Since v can be found using one pass of regular DP, we can find v for each column in $O(mn)$ time and linear space since we don't need to keep track of traceback pointers for this step. Then by applying the divide and conquer approach, the subproblems take half the time since we only need to keep track of the cells diagonally along the optimal alignment path (half of the matrix of the previous step) That gives a total run time of $O(mn(1 + \frac{1}{2} + \frac{1}{4} + \ldots)) = O(2MN) = O(mn)$ (using the sum of geometric series), to give us a quadratic run time (twice as slow as before, but still same asymptotic behavior). The total time will never exceed $2MN$ (twice the time as the previous algorithm). Although the run-time is increased by a constant factor, one of the big advantages of the divide-and-conquer approach is that the space is dramatically reduced to $O(N)$.

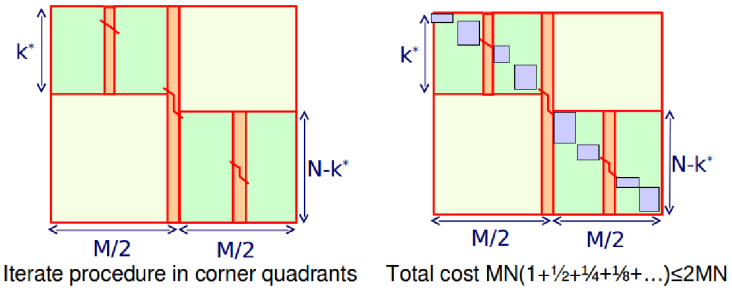


Figure 3.9: Divide and Conquer

3.2.3 Generalized gap penalties

Gap penalties determine the score calculated for a subsequence and thus determine which alignment is selected. Depending on the model, it could be a good idea to penalize differently for, say, gaps of different lengths. However, the tradeoff is that there is also cost associated with using more complex gap penalty functions by substantially increasing running time. This cost can be mitigated by using simpler approximations to the gap penalty functions. The affine function is a fine intermediate: you have a fixed penalty to start a gap then a linear cost to add to a gap. You can also consider more complex functions that take into consideration the properties of protein coding sequences. In the case of protein coding regions alignment, a gap of length mod 3 can be less penalized.

3.3 Linear-time exact string matching

While we have looked at various forms of alignment and algorithms used to find such alignments, these algorithms are not fast enough for practical purposes. E.g. we may have a 100 nucleotide sequence which we want to search for in the whole genome, which may be over a billion nucleotides long. In this case, we want an algorithm with a run-time that depends on the length of query sequence, possibly with some pre-processing on the database, because processing the entire genome for every query would be extremely slow. For such problems, we enter the realm of randomized algorithms where instead of worrying about the worst-case performance, we are more interested in making sure that the algorithm is linear in the expected case. When looking for exact(consecutive) matches of a sequence, the Karp-Rabin algorithm interprets such a match numerically. There are many other solutions to this problem and some of them that can ensure the problem is linear in the worst case such as: the Z-algorithm, Boyer-Moore and Knuth-Morris-Pratt algorithm, algorithms based on suffix trees, suffix arrays, etc. (discussed in the "Lecture 3 addendum" slides)

3.3.1 Karp-Rabin Algorithm

This algorithm tries to match a particular pattern to a string this is the basic principle of database search. The problem is as follows: in text T of length n we are looking for pattern P of length m. The key idea of the algorithm is that strings are mapped to numbers to enable fast comparison. A nave version of the algorithm involves mapping the string P and substrings of T of length m into numbers x and y, respectively, sliding x along T at every offset until there is a match of the numbers.

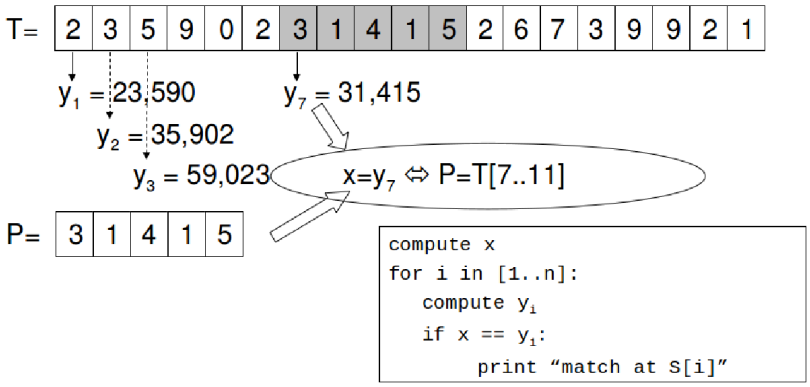


Figure 3.10: Naive Karp-Rabin algorithm

However, one can see that the algorithm, as stated, is in fact non-linear for two reasons:

1. Computing each y_i takes more than constant time (it is in fact linear if we naively compute each number from scratch for each subsequence)

2. Comparing x and y_i can be expensive if the numbers are very large which might happen if the pattern to be matched is very long

To make the algorithm faster, we first modify the procedure for calculating y_i in constant time by using the previously computed number, $y_i - 1$. We can do this using some bit operations: a subtraction to remove the high-order bit, a multiplication to shift the characters left, and an addition to append the low-order digit.

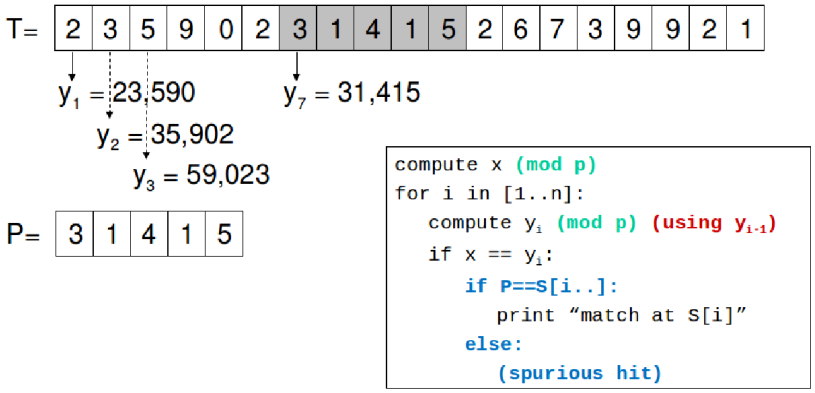


Figure 3.11: Efficiently converting strings to numbers

Also, to keep the numbers small to ensure efficient comparison, we do all our computations modulo p (a form of hashing), where p reflects the word length available to us for storing numbers. At the same time, p is chosen in such a way that the comparison between x and y_i is not expensive. However, mapping to the space of numbers modulo p can result in spurious hits due to hashing collisions, and so we modify the algorithm to deal with such spurious hits by explicitly verifying reported hits of the hash values. Hence, the final version of the Karp-Rabin algorithm is:

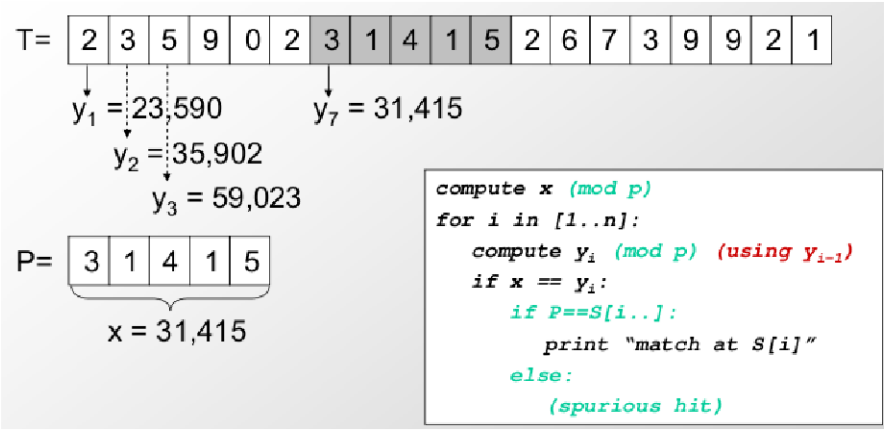


Figure 3.12: Final Karp-Rabin algorithm

Questions:

Q4: What if there are more than 10 characters in the alphabet?

A4: In such a case, we can just modify the above algorithm by including more digits i.e. by working in a base other than 10, e.g. say base 256. But in general, when hashing is used, strings are mapped into a space of numbers and hence the strings are interpreted numerically.

Q5: Are there provisions in the algorithm for inexact matches?

A5: True, the above algorithm only works when there are regions of exact similarity between the query sequence and the database. In fact, the BLAST algorithm, which we look at later, extends the above

ideas to include the notion of searching in a biologically meaningful neighborhood of the query sequence to account for some inexact matches. This is done by searching in the database for not just the query sequence, but also some variants of the sequence up to some fixed number of changes.

In general, in order to reduce the time for operations on arguments like numbers or strings that are really long, it is necessary to reduce the number range to something more manageable. Hashing is a general solution to this and it involves mapping keys k from a large universe U of strings/numbers into a hash of the key $h(k)$ which lies in a smaller range, say $[1...m]$. There are many hash function that can be used, all with different theoretical and practical properties. The two key properties that we need are:

1. Reproducibility if $x = y$, then $h(x) = h(y)$. This is essential for our mapping to make sense

2. Uniform output distribution This implies that regardless of the input distribution, the output distribution is uniform. i.e. if $x! = y$, then $P(h(x) = h(y)) = 1/m$, irrespective of the input distribution. This is a desirable property to reduce the chance of spurious hits.

An interesting idea that was raised was that it might be useful to have locality sensitive hash functions from the point of view of use in neighborhood searches, such that points in U that are close to each other are mapped to nearby points by the hash function. The notion of Random projections, as an extension of the BLAST algorithm, is based on this idea. Also, it is to be noted that modulo doesnt satisfy property 2 above because it is possible to have input distributions (e.g. all multiples of the number vis--vis which the modulo is taken) that result in a lot of collisions. Nevertheless, choosing a random number as the divisor of the modulo can avoid many collisions. Still, working with hashing increases the complexity of analyzing the algorithm since now we need to compute the expected run time by including the cost of verification. To show that the expected run time is linear, we need to show that the probability of spurious hits is small.

3.4 The BLAST algorithm (Basic Local Alignment Search Tool)

The BLAST algorithm looks at the problem of sequence database search, wherein we have a query, which is a new sequence, and a target, which is a set of many old sequences, and we are interested in knowing which (if any) of the target sequences is the query related to. One of the key ideas of BLAST is that it does not require the individual alignments to be perfect once an initial match is identified, we can fine-tune the matches later to find a good alignment. Also, BLAST exploits a distinct characteristic of database search problems that most target sequences will be completely unrelated to the query sequence, and very few sequences will match. What this means is that correct (near perfect) alignments will have long substrings of nucleotides that match perfectly. E.g. if we looking for sequences of length 100 and are going to reject matches that are less than 90% identical, we need not look at sequences that do not even contain a consecutive stretch of 10 matching nucleotides in a row. In biology, the mutations that we find will not actually be distributed randomly, but will be clustered in nonfunctional regions of DNA while leaving long stretches of functional DNA untouched. Therefore since highly similar sequences will have stretches of similarity, we can pre-screen the sequences for common long stretches this idea is used in BLAST by breaking up the query sequence into W-mers and pre-screening the target sequences for all possible $W - mers$.

The other aspect of BLAST that allows us to speed up repeated queries is the ability to preprocess a large database of DNA off-line. After preprocessing, searching for a sequence of length m in a database of length n will take only $O(m)$ time. The key insights that BLAST is based on are the ideas of hashing and neighborhood search that allows one to search for $W - mers$, even when there are no exact-matches.

3.4.1 The BLAST algorithm

The steps are as follows:

1. Split query into overlapping words of length W (the W-$mers$

2. Find a "neighborhood" of similar words for each word (see below)

3. Lookup each word in the neighborhood in a hash table to find the location in the database where each word occurs. Call these the *seeds*, and let S be the collection of seeds.

4. Extend the seeds in S until the score of the alignment drops off below some threshold X.

5. Report matches with overall highest scores

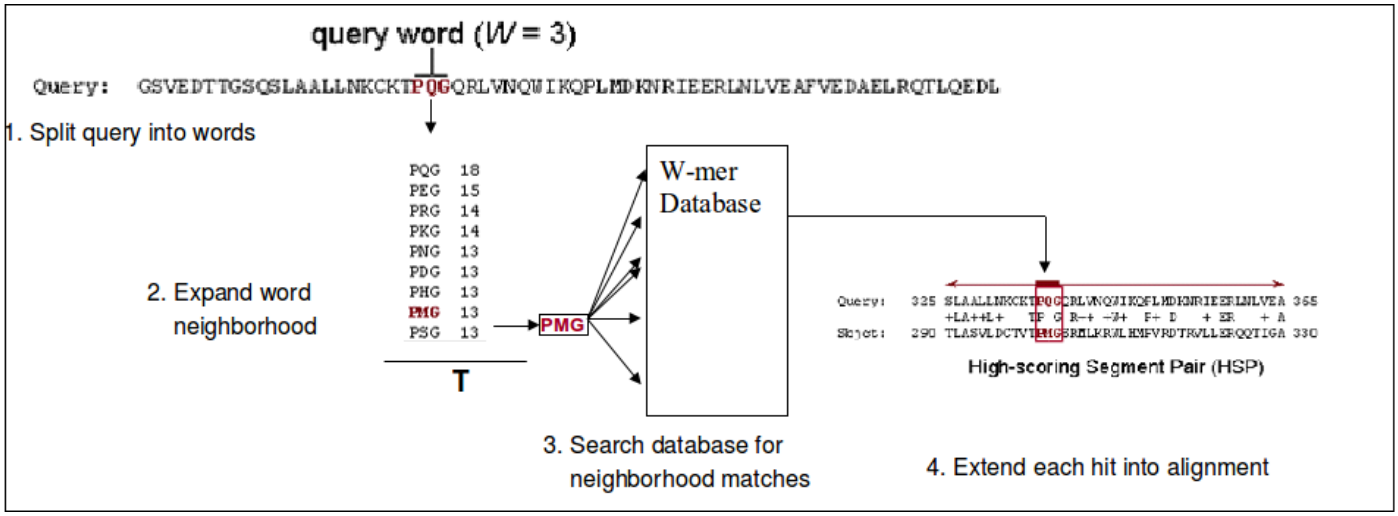


Figure 3.13: The BLAST Algorithm

The pre-processing step of BLAST is to make sure that all substrings of W nucleotides will be included in our database (or in a hash table). These are called the *W-mers* of the database. As in step 1, we first split the query by looking at all substrings of W consecutive nucleotides in the query. To find the neighborhood of these *W-mers*, we then modify these sequences by changing them slightly and computing their similarity to the original sequence. We generate progressively more dissimilar words in our neighborhood until our similarity measure drops below some threshold T. This affords us to flexibility to find matches that do not have exactly W consecutive matching characters in a row, but which do have enough matches to be considered similar.

Then, we look up all of these words in our hash table to find seeds of W consecutive matching nucleotides. We then extend these seeds to find our alignment using the Smith-Waterman algorithm for local alignment, until the score drops below a certain threshold. Since the region we are considering is a much shorter segment, this will not be as slow as running the algorithm on the entire DNA database.

It is also interesting to note the influence of various parameters of BLAST on the performance of the algorithm vis--vis run-time and sensitivity:

- **W** Although large W would result in fewer spurious hits/collisions, thus making it faster, there are also tradeoffs associated, namely: a large neighborhood of slightly different query sequences, a large hash table, and too few hits. On the other hand, if W is too small, we may get too many hits which might make the algorithm slower.

- **T** If T is higher, the algorithm will be faster, but you may miss sequences that are more evolutionarily distant. If comparing two related species, you can probably set a higher T since you expect to find more matches between sequences that are quite similar.

- **X** Its influence is quite similar to T in that both will control the sensitivity of the algorithm. While W and T affect the total number of hits one gets, and hence affect the run-time of the algorithm dramatically, setting a really stringent X despite the more sensitive W and T, will result in the algorithm being less sensitive and slower. So, it is important to match the stringency of X with that of W and T.

3.4.2 Extensions to BLAST

- **Filtering** Low complexity regions can cause spurious hits. For instance, if our query has a string of copies of the same nucleotide e.g. repeats of AC or just G, and the database has a long stretch of the same nucleotide, then there will be many many useless hits. To prevent this, we can either try to filter out low complexity portions of the query or we can ignore unreasonably over-represented portions of the database.

- **Two-hit BLAST** The idea here is to use double hashing wherein instead of hashing one long W-*mer*, we will hash two small W-mers. This allows us to find small regions of similarity since it is much more likely to have two smaller W-*mers* that match rather than one long W-*mer*. This allows us to get a higher sensitivity with a smaller W, while still pruning out spurious hits. This means that well spend less time trying to extend matches that dont actually match. Thus, this allows us to improve speed while maintaining sensitivity.

  Q6: For a long enough W, would it make sense to consider more than 2 smaller W-*mers*?

  A6: Before, it would be interesting to see how the number of such W-*mers* influences the sensitivity of the algorithm.

- **Combs** This is the idea of using non-consecutive W-*mers* for hashing. Recall from your biology classes that the third nucleotide in a triplet usually doesnt actually have an effect on which amino acid is represented. This means that each third nucleotide in a sequence is less likely to be preserved by evolution, since it often doesnt matter. Thus, we might want to look for W-*mers* that look similar except in every third codon. This is a particular example of a comb. A comb is simply a bit mask which represents which nucleotides we care about when trying to find matches. We explained above why 110110110 . . . might be a good comb, and it turns out to be. However, other combs are also useful. One way to choose a comb is to just pick some nucleotides at random. Rather than picking just one comb for a projection, it is possible to randomly pick a set of such combs and project the W-mers along each of these combs to get a set of lookup databases. Then, the query string can also be projected randomly along these combs to lookup in these databases, thereby increasing the probability of finding a match. This is called Random Projection. Extending this, an interesting idea for a final project is to think of different techniques of projection or hashing that make sense biologically.

3.5 Pre-processing for linear-time string matching

The hashing technique at the core of the BLAST algorithm is a powerful way of string for rapid lookup. A substantial time is invested to process the whole genome, or a large set of genomes, *in advance* of obtaining a query sequence. Once the query sequence is obtained, it can be similarly processed and its parts searched against the indexed database in linear time.

In this section, we briefly describe four additional ways of pre-processing a database for rapid string lookup, each of which has both practical and theoretical importance.

3.5.1 Suffix Trees

Suffix trees provide a powerful tree representation of substrings of a target sequence T, by capturing all suffixes of T in a radix tree.

Representation of a sequence in a suffix tree

TODO: EXPAND *@scribe: Add image of suffix tree representation*

Searching a new sequence against a suffix tree

TODO: EXPAND *@scribe: Describe search procedure*

Linear-time construction of suffix trees

TODO: EXPAND *@scribe: The three key ideas for linear-time construction*

3.5.2 Suffix Arrays

For many genomic applications, suffix trees are too expensive to store in memory, and more efficient representations were needed to do so. Suffix arrays were developed specifically to reduce the memory consumption of suffix trees, and achieve the same goals with a significantly reduced space need.

TODO: EXPAND *@scribe: Add informatio on suffix arrays*

Using suffix arrays, any substring can be found by doing a binary search on the ordered list of suffixes. By thus exploring the prefix of every suffix, we end up searching all substrings.

3.5.3 The Burrows-Wheeler Transform

An even more efficient representation than suffix trees is given by the Burrows-Wheeler Transform (BWT), which enables storing the entire hashed string in the same number of characters as the original string (and even more compactly, as it contains frequent homopolymer runs of characters that can be more easily compresed). This has helped make programs that can run even more efficiently.

We first consider the BWT matrix, which is an extension of a suffix array, in that it contains not only all suffixes in sorted (lexicographic) order, but it appends to each suffix starting at position i the prefix ending at position $i-1$, each row thus containing a full rotation of the original string. This enables all the suffix-array and suffix-tree operations, of finding the position of suffixes in time linear in the query string.

The key difference from Suffix Arrays is space usage, where instead of storing all suffixes in memory, which even for suffix arrays is very expensive, only the last column of the BWT matrix is stored, based on which the original matrix can be recovered.

TODO: EXPAND *@scribe: Add example of compress-uncompress*

An auxiliary array can be used to speed things even further and avoid having to repeat operations of finding the first occurrence of each character in the modified suffix array.

Lastly, once the positions of 100,000s of substrings are found in the modified string (the last column of the BTW matrix), these coordinates can be transformed to the original positions, saving runtime by amortizing the cost of the transformation across the many many reads.

The BWT has had a very strong impact on short-string matching algorithms, and nearly all the fastest read mappers are currently based on the Burrows-Wheeler Transform.

3.5.4 Fundamental pre-processing

A variation of processing that has theoretical interest but has found relatively little practical use in bioinformatics. It relies on the Z vector, that contains at each position i the length of the longest prefix of a string that also matches the substring starting at i. This enables computing the L and R (Left and Right) vectors that denote the end of the longest duplicate substrings that contains the current position i.

TODO: EXPAND *@scribe: Use lecture from* `http://courses.csail.mit.edu/6.006/spring11/lectures/lec18-extra.pdf` *to expand this section*

The Z algorithm enables an easy computation of both the Boyer-Moore and the Knuth-Morris-Pratt algorithms for linear-time string matching.

3.6 Probabilistic Foundations of Sequence Alignment

As described above, the BLAST algorithm uses a scoring (substitution) matrix to expand the list of W-*mers* to look for and to determine an approximately matching sequence during seed extension. Also, a scoring matrix is used in evaluating matches or mismatches in the alignment algorithms. But how do we construct this matrix in the first place? How do you determine the value of $s(x_i, y_j)$ in global/local alignment?

	A	G	T	C
A	+1	-½	-1	-1
G	-½	+1	-1	-1
T	-1	-1	+1	-½
C	-1	-1	-½	+1

Figure 3.14: Nucleotide and amino acid match scores

The idea behind the scoring matrix is that the score of alignment should reflect the probability that two similar sequences are homologous i.e. the probability that two sequences that have a bunch of nucleotides in common also share a common ancestry. For this, we look at the likelihood ratios between two hypotheses.

1. **Hypothesis 1:** – That the alignment between the two sequence is due to chance and the sequences are, in fact, unrelated.
2. **Hypothesis 2:** – That the alignment is due to common ancestry and the sequences are actually related.

Then, we calculate the probability of observing an alignment according to each hypothesis. $Pr(x, y|U)$ is the probability of aligning x with y assuming they are unrelated, while $Pr(x, y|R)$ is the probability of the alignment, assuming they are related. Then, we define the alignment score as the log of the likelihood ratio between the two:

$$S \equiv \log \frac{P(\mathbf{x}, \mathbf{y}|R)}{P(\mathbf{x}, \mathbf{y}|U)} \tag{3.1}$$

So, when we add up the scores of individual alignments at each position in the alignment of two sequences, since a sum of logs is a log of a product, we get that the total score of the alignment gives the probability of the whole alignment, assuming each individual alignment is independent. Thus, an additive matrix score exactly gives us the probability that the two sequences are related, and the alignment is not due to chance. More formally, considering the case of aligning proteins, for unrelated sequences, the probability of having an n-residue alignment between x and y is a simple product of the probabilities of the individual sequences since the residue pairings are independent. That is,

$$\begin{aligned} \mathbf{x} &= \{x_1 \dots x_n\} \\ \mathbf{y} &= \{y_1 \dots x_n\} \\ q_a &= P(\text{amino acid } a) \\ P(\mathbf{x}, \mathbf{y}|U) &= \prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{n} q_{y_i} \end{aligned}$$

For related sequences, the residue pairings are no longer independent so we must use a different joint probability, assuming that each pair of aligned amino acids evolved from a common ancestor:

$$\begin{aligned} p_{ab} &= P(\text{evolution gave rise to } a \text{ in } \mathbf{x} \text{ and } b \text{ in } \mathbf{y}) \\ P(\mathbf{x}, \mathbf{y}|R) &= \prod_{i=1}^{n} p_{x_i y_i} \end{aligned}$$

Then, the likelihood ratio between the two is given by:

$$\begin{aligned}\frac{P(\mathbf{x},\mathbf{y}|R)}{P(\mathbf{x},\mathbf{y}|U)} &= \frac{\prod_{i=1}^{n} p_{x_i y_i}}{\prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{n} q_{y_i}} \\ &= \frac{\prod_{i=1}^{n} p_{x_i y_i}}{\prod_{i=1}^{n} q_{x_i} q_{y_i}}\end{aligned}$$

Since we eventually want to compute a sum of scores and probabilities require add products, we take the log of the product to get a handy summation:

$$\begin{aligned}S &\equiv \log \frac{P(\mathbf{x},\mathbf{y}|R)}{P(\mathbf{x},\mathbf{y}|U)} \\ v &= \sum_i \log\left(\frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}\right) \\ &\equiv \sum_i s(x_i, y_i)\end{aligned} \tag{3.2}$$

Thus, the substitution matrix score for a given pair a, b is give by

$$s(a,b) = \log\left(\frac{p_{ab}}{q_a q_b}\right)$$

The above expression is then used to crank out a substitution matrix like the BLOSUM62 for amino acids. It is interesting to note that the score of a match of an amino acid with itself depends on the amino acid itself because the frequency of random occurrence of an amino acid affects the terms used in calculating the likelihood ratio score of alignment. Hence, these matrices capture not only the sequence similarity of the alignments, but also the chemical similarity of various amino acids.

3.7 Current Research Directions

3.8 Further Reading

3.9 Tools and Techniques

3.10 What Have We Learned?

Bibliography

CHAPTER

FOUR

COMPARATIVE GENOMICS I: GENOME ANNOTATION

Mark Smith, Yarden Katz
(Partially adapted from notes by:
Angela Yen, Christopher Robde, Timo Somervuo and Saba Gul)

Figures

4.1 Introduction

In this chapter we will explore the emerging field of comparative genomics, primarily through examples of multiple species genome alignments (work done by the Kellis lab.) One approach to the analysis of genomes is to infer important gene functions using an understanding of evolution to search for expected evolutionary patterns. Another approach is to discover evolutionary trends by studying genomes themselves. Taken together, evolutionary insight and large genomic datasets offer great potential for discovery of novel biological phenomena.

A recurring theme of this work is that by taking a global computational approach to analyzing elements of genes and RNAs encoded in the genome, one can find interesting new biological phenomena by seeing how individual examples "diverge" or differ from the average case. For example, by examining many protein–coding genes, we can identify features representative of that class of loci, and then come up with highly accurate tests for distinguishing protein–coding from non–protein–coding genes. Often, these computational tests, based on thousands of examples, will be far more definitive than conventional low–throughput wet lab tests (such as mass spectrometry to detect protein products, in cases where we want to know if a particular locus is protein coding.)

4.1.1 Motivation and Challenge

As the cost of genome sequencing continues to drop, the availability of sequenced genome data has exploded. However, analysis of the data has not kept up, and it is likely that there are many interesting biological phenomena lying undiscovered in the endless strings of ATGCs. The goal of comparative genomics is to leverage the vast amounts of information available to look for biological patterns.

As the name suggests, comparative genomics involves the simultaneous comparison of genomes from many species which evolved from a common ancestor. As the hand of evolution guides changes to a species's genome, it leaves behind traces of its presence. We will see later in this chapter that evolution discriminates between portions of a genome on the basis of biological function. By exploiting this correlation between evolution's fingerprints on and the biological role of a genomic subsequence, comparative genomics is able to direct wet lab research to interesting portions of the genome and even discover new biological phenomena.

For example, CRISPRs, clustered regularly interspaced short palindromic repeats, which are found in bacteria and archaea, were first discovered by the methods of comparative genomics. Follow–up experiments revealed that they provide adaptive immunity to plasmids and phages. Later in this chapter, we will look at the phenomenon of stop–codon read–through, where stop codons are occasionally ignored during the process of translation phase of protein biosynthesis. Without comparative genomics to guide them, experimentalists might have ignored both of these features for many years.

Without a system for interpreting and identifying important features in genomes, all of the DNA sequences on earth are just a meaningless sea of data. An approach naive to biology might miss the signatures of synonymous substitutions or frame shift mutations, while an approach naive to computer science might be hopelessly inefficient when faced with the ever larger datasets emerging from sequencing centers. Comparative genomics requires rare multidisciplinary skills and insight.

This is a particularly exciting time to enter the field of comparative genomics, because the field is mature enough that there are tools and data available to make discoveries but young enough that important findings will likely continue to be made for many years.

4.1.2 Importance of many closely–related genomes

In order to resolve significant biological features we need both sufficient similarity to enable comparison and sufficient divergence to identify signatures of change over evolutionary time, which are difficult to achieve in a pairwise comparison. We improve the resolution of our analysis by extending analysis to many genomes simultaneously with some clusters of similar organisms and some dissimilar organisms. A simple analogy is one of observing an orchestra. If you place a single microphone, it will be difficult to decipher the signal coming from the entire system, because it will be overwhelmed by the local noise from the single point of observation, the nearest instrument. If you place many microphones distributed across the orchestra at reasonable distances, then you get a much better perspective not only on the overall signal, but also on the structure of the local noise. Similarly, by sequencing many genomes across the tree of life we are able to distinguish the biological signals of functional elements from the noise of neutral mutations. This is because nature selects for conservation of functional elements across large phylogenetic distances while constantly introducing noise through mutagenic processes operating at shorter time scales.

In this chapter, we will assume that we already have a complete genome–wide alignment of multiple closely–related species, spanning both coding and non–coding regions. In practice, constructing complete genome assemblies and whole–genome alignments is a very challenging problem; that will be the topic of the next chapter.

4.2 Conservation of genomic sequences

4.2.1 Functional elements in *Drosophila*

In a 2007 paper[1], Stark et al. identified evolutionary signatures of different functional elements and predicted function using conserved signatures. One important finding is that across evolutionary time, genes tend to remain in a similar location. This is illustrated by figure 4.1, which shows the result of a multiple alignment on orthologous segments of genomes from twelve *Drosophila* species. Each genome is represented by a horizontal blue line, where the top line represents the reference sequence. Grey lines connect orthologous functional elements, and it is clear that their positions are generally conserved across the different species.

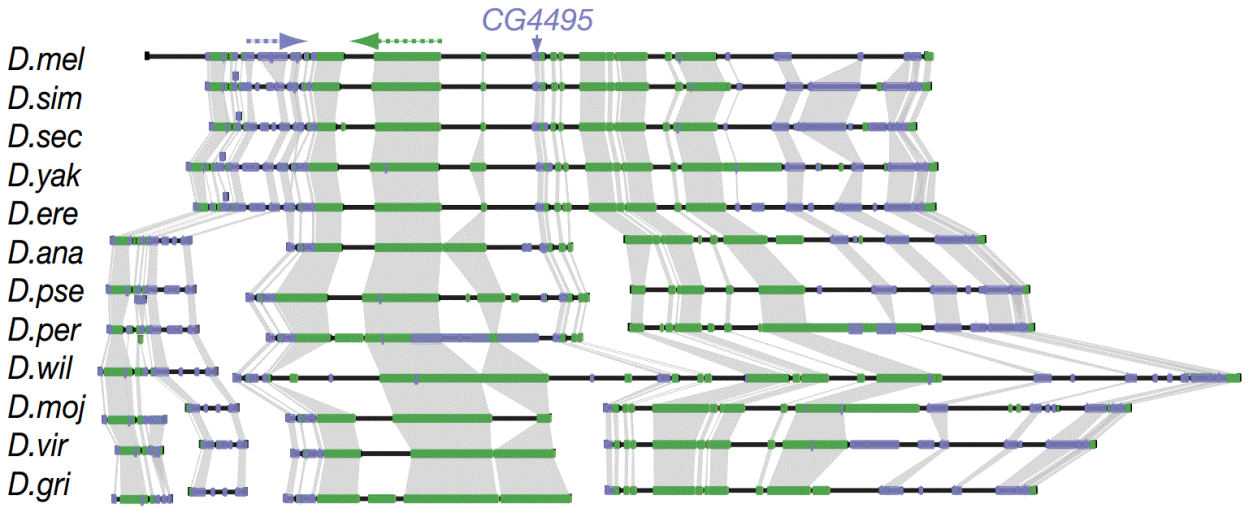


Figure 4.1: Comparative identification of functional elements in 12 *Drosophila* species. Grey lines indicate the alignment of orthologous regions. Color indicates direction of transcription.

4.2.2 Rates and patterns of selection

Now that we have established that there is structure to the evolution of genomic sequences, we can begin analyzing specific features of the conservation. For this section, let us consider genomic data at the level of individual nucleotides. Later on in this chapter we will see that we can also analyze amino acid sequences.

One property which we may consider is the rate of nucleotide substitution in a genome. Figure 4.2 shows two nucleotide sequences from a collection of mammals. One of the sequences is subject to normal rates of change, while the other demonstrates a reduced rate. Hence we may hypothesize that the latter sequence is subject to a greater level of evolutionary constraint, and may represent a more biologically important section of the genome.

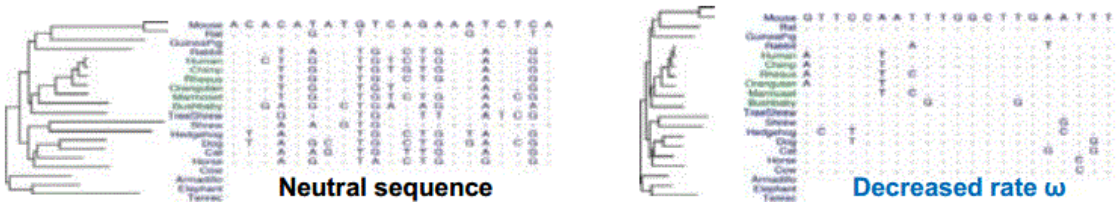


Figure 4.2: A comparison between two genomic regions with different selection rates ω. The sequence on the left demonstrates normal rates of mutation, while the sequence on the right shows a high conservation level, as evidenced by the reduced number of mutations.

[1] http://www.nature.com/nature/journal/v450/n7167/abs/nature06340.html

Independently of the substitution rate, we may also consider the pattern of substitutions in a particular nucleotide subsequence. Consider a sequence of nucleotides which encodes a protein. Due to tRNA wobble, a mutation in the third nucleotide of a codon is less likely to affect the final protein than a mutation in the other positions. Hence we expect to see a pattern of increased substitutions on the third position when looking at protein–coding subsequences of the genome. This is indeed verified experimentally, as shown in figure 4.3.

Figure 4.3: Different mutation patterns in protein–coding and non–protein–coding regions. Asterisks indicate that the nucleotide was conserved across all species. Note that within the protein–coding exon, nucelotides 1 and 2 of each codon tend to be conserved, while codon 3 is allowed to vary more, which is consistent with the phenomenon of wobble.

4.3 Excess Constraint

In most regions of the genome where we see conservation across species, we expect there to be at least some amount of *synonymous* substitution. These are "silent" nucleotide substitutions that modify a codon in such a way that the amino acid it encodes is unchanged. In a 2011 paper[2], Lindblad–Toh et al. studied evolutionary constraint in the human genome by doing comparative analysis of 29 mammalian species. They found that among the 29 genomes, the average nucleotide site showed 4.5 substitutions per site.

Given such a high average substitution rate, we do not expect to see perfect conservation across all regions that are conserved. For example, ignoring all other effects, the probability of a 12–mer remaining fixed across all 29 species is less than 10^{-25}. Thus, regions which are nearly perfect conserved across multiple species stand out as being unique and worthy of further study. One such region is shown in figure 4.4.

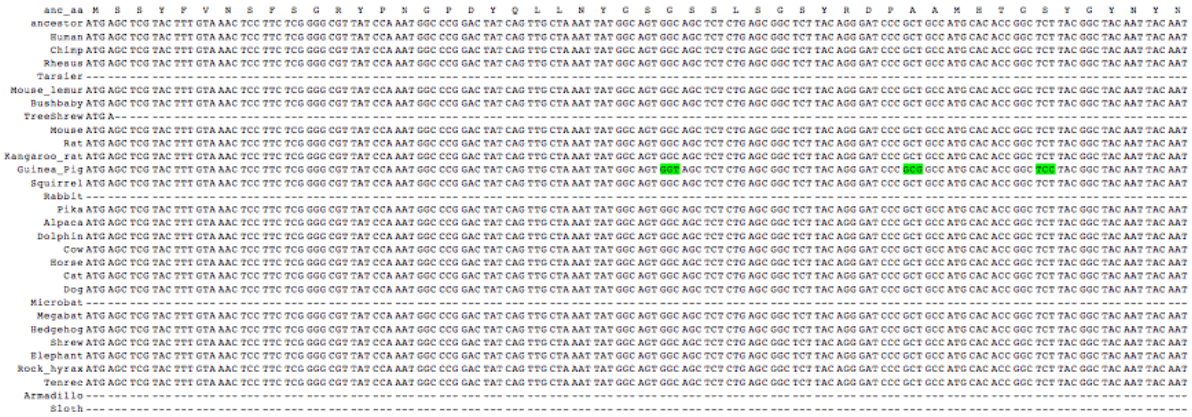


Figure 4.4: Many genomic regions, such as HOXB5, show more conservation than we would expect in normal conserved coding regions. Among the 29 species under study, all but 7 of them had the exact same nucleotide sequence.

[2] http://www.nature.com/nature/journal/v478/n7370/full/nature10530.html

4.3.1 Causes of Excess Constraint

The question is what evolutionary pressures cause certain regions to be so perfectly conserved? The following were all mentioned in class as possibilities:

- Could it be that there is a special structure of DNA shielding this area from mutation?
- Is there some special error correcting machinery that sits at this spot?
- Can the cell use the methylation state of the two copies of DNA as an error correcting mechanism? This mechanism would rely on the fact that the new copy of DNA is unmethylated, and therefore the DNA replication machinery could check the new copy against the old methylated copy.
- Maybe the next generation can't survive if this region is mutated?

Another possible explanation is that selection is occurring to conserve specific codons. Some codons are more efficient than others: for example, higher abundant proteins that need rapid translation might select codons that give the most efficient translation rate, while other proteins might select for codons that give less efficient translation.

Still, these regions seem too perfectly conserved to be explained by codon usage alone. What else can explain excess constraint? There must be some degree of accuracy needed at the nucleotide level that keeps these sequences from diverging.

It could be that we are looking at the same region in two species that have only recently diverged or that there is a specific genetic mechanism protecting this area. However, it is more likely that so much conservation is a sign of protein coding regions that simultaneously encode other functional elements. For example, the HOXB5 gene shows obvious excess constraint, and there is evidence that the 5' end of the HOXB5 ORF encodes both protein and an RNA secondary structure.

Regions that encode more than one type of functional element are under overlapping selective pressures. There might be pressure in the protein coding space to keep the amino acid sequence corresponding to this region the same, combined with pressure from the RNA space to keep a nucleotide sequence that preserves the RNA's secondary structure. As a result of these two pressures to keep codons for the same amino acids and to produce the same RNA structure, the region is likely to show much less tolerance for any synonymous substitution patterns.

4.3.2 Modeling Excess Constraint

To better study region of excess constraint, we develop mathematical models to systematically measure the amount of synonymous and nonsynonymous conservation of different regions. We will measure two rates: codon and nucleotide conservation.

For looking at substitutions at both the nucleotide and codon levels, we need to define the null model, where synonymous substitutions occur at some rate, and then define the alternative model that we can compare to the null model and get a measure of the amount of excess constraint.

To represent the null model, we can build rate matrices (4×4 in the nucleotide case and 64×64 for the codon case) that give the rates of substitutions between either codons or nucleotides for a unit time. We estimate the rates in the null model by looking at a ton of data and estimating the probabilities of each type of substitution. See figure 4.13a in §4.5.2 for an example of a null matrix for the codon case.

For the alternative model, we can define *scaling factors* in each region. These scaling factors measure how the rate of each type of substitution in our region of interest compares to that of the null model.

- λ_s: the rate of synonymous substitutions
- λ_n: the rate of nonsynonymous substitutions

For example, if $\lambda_s = 0.5$, then the rate of synonymous substitutions is half of what is expected from the null model in that region. We can then evaluate the statistical significance of the rate estimates we obtain, and find regions where the rate of substitution is much lower than expected.

Using a null model here helps us account for biases in alignment coverage of certain codons and also accounts for the possibility of codon degeneracy, in which case we would expect to see a much higher rate of substitutions. We will learn how to combine such models with phylogenic methods when we talk about phylogenic trees and evolution later on in the course.

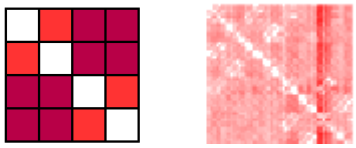


Figure 4.5: We can model mutations using rate matrices, as shown here for nucleotide substitutions on the left and codon substitutions on the right. In each matrix, the cell in the mth row and nth column represents the likelihood that the mth symbol will mutate into the nth symbol. The darker the color, the less likely the mutation.

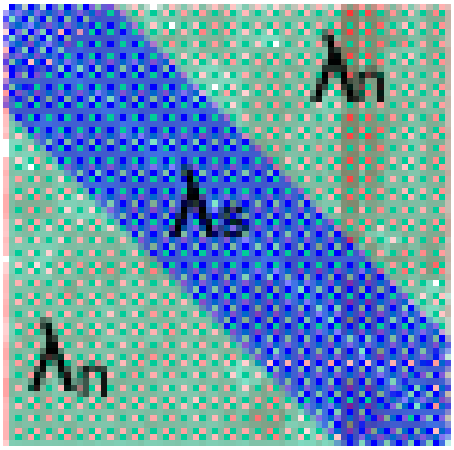


Figure 4.6: We can measure λ_s and λ_n, which give the rate of synonymous and nonsynonymous substitutions, respectively, in a given region.

Applying this model shows that the sequences in the first translated codons, cassette exons (exons that are present in one mRNA transcript but absent in an isoform of the transcript), and alternatively spliced regions have especially low rates of synonymous substitutions.

4.3.3 Excess Constraint in the Human Genome

In this section, we will examine the problem of determining the total proportion of the human genome under excess constraint. In particular, we will revisit the work of Lindblad–Toh et al. of comparing 29 mammalian genomes. They measured conservation levels throughout the genome by applying the process described in the previous section to 50–mers. By considering only 50–mers which were part of ancestral repeats, it is possible to determine a background level of conservation. We can imagine that the intensities of conservation among the 50–mers are distributed according to a hidden probability distribution, as illustrated in figure 4.7. In the figure, the background curve represents the distribution of constraint in the absence of special mechanisms for excess constraint, as determined by looking at ancestral repeats, while the signal (foreground) curve represents the actual distribution of the genome. The signal curve has more conservation overall due to the effects of purifying selection.

We may wish to investigate specific regions of the genome which are under excess constraint by setting a threshold level of conservation and examining regions which are more conserved. In the illustration, this corresponds to considering all 50–mers which fall to the right of one of the orange line. We see that while this method does indeed give us regions under excess constraint, it also gives us false positives. This is because even in the absence of purifying selection and other effects, certain regions will be heavily conserved, simply due to random chance. Setting the threshold higher, such as by using the dotted orange line as our threshold, reduces the proportion of false positives to true positives, while also lowering the number of true

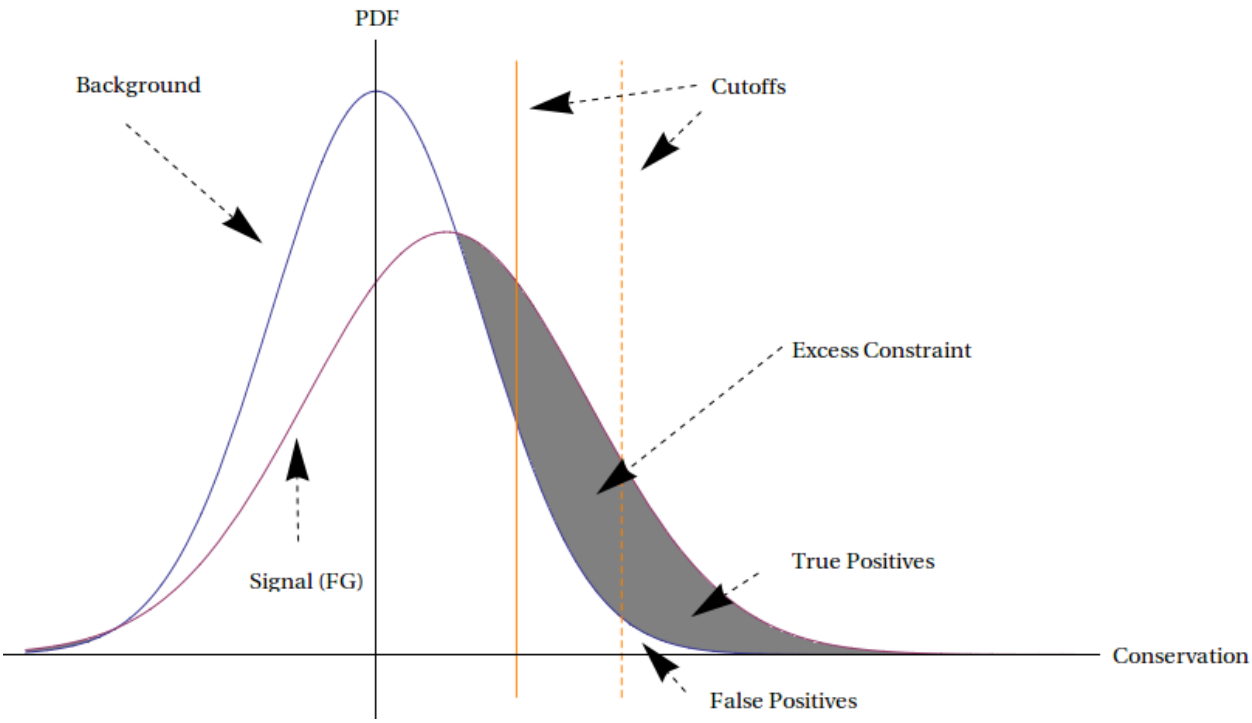


Figure 4.7: Measuring genome–wide excess constraint. See accompanying text for explanation.

positives detected, thus trading higher specificity for lower sensitivity.

However, not all hope is lost. It is possible to empirically measure both the background and signal curves, as described above. Once that is done, the area of the region between them, which is shaded in gray in figure 4.7, can be determined by integration. This area represents the proportion of the genome which is under excess constraint. This number turns out to be about 5% of the human genome, depending on how large a window is used. Those regions are likely to all be functional, but since about 1.5% of the human genome is protein–coding, we can infer that the remaining 3.5% consists of functional, non–coding elements, most of which probably play regulatory roles.

4.3.4 Examples of Excess Constraint

Examples of excess constraint have been found in the following cases:

- Most Hox genes show overlapping constraint regions. In particular, as mentioned above the first 50 amino acids of HOXB5 are almost completely conserved. In addition, HOXA2 shows overlapping regulatory modules. These two loci encode developmental enhancers, providing a mechanism to provide tissue specific expression.

- ADAR: the main regulator of mRNA editing, has a splice variant where a low synonymous substitution rate was found at a resolution of 9 codons.

- BRCA1: Hurst and Pal (2001) found a low rate of synonymous substitutions in certain regions of BRCA1, the main gene involved in breast cancer. They hypothesized that purifying selection is occurring in these regions. (This claim was refuted by Schmid and Yang (2008) who claim this phenomenon is the artifact of a sliding window analysis).

- THRA/NR1D1: these genes, also involved in breast cancer, are part of a dual coding region that codes for both genes and is highly conserved.

- SEPHS2: has a hairpin involved in selenocysteine recoding. Because this region must select codons to both conserve the protein's amino acid sequence and the nucleotides to keep the same RNA secondary structure, it shows excess constraint.

4.4 Diversity of evolutionary signatures: An Overview of Selection Patterns

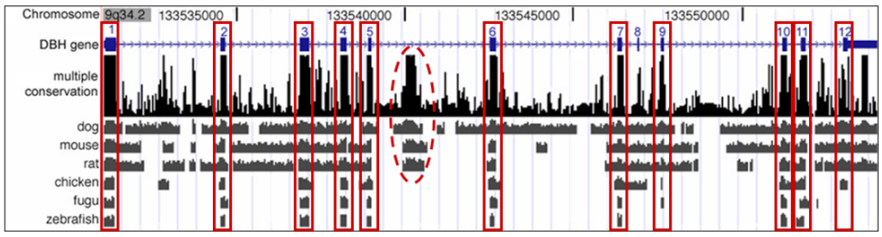


Figure 4.8: Exons (boxed in red) are deeply conserved from mammals to fish. Other elements are also strongly conserved, such as the circled peak near the center of the graph. This may be a regulatory element found in mammals but not in aves or fish.

In the figure above, we see a DNA sequence represented on the x–axis, while each "row" represents a different species. The y–axis within each row represents the amount of conservation for that species in that part of the chromosome (though other species that are not shown were also used to calculate conservation). To calculate the degree of conservation, a Hidden Markov Model (HMM)[3] was used with two states: high conservation and low conservation. The amount of conservation represents the probability of being in the conserved state of the model at this locus, derived from the score calculated using posterior decoding of the high conservation model.

From this figure, we can see that there are blocks of conservation separated by regions that are not conserved. The 12 exons (highlighted by red rectangles) are mostly conserved across species, but sometimes, certain exons are missing; for example, zebrafish is missing exon 9. However, we also see that there is a spike in some species (as circled in red) that do not correspond to a known protein coding gene. This tells us that some intronic regions have also been evolutionarily conserved, since DNA regions that do not code for proteins can still be important as functional elements, such as RNA, microRNA, and regulatory motifs. By observing how regions are conserved, instead of just looking at the amount of conservation, we can observe 'evolutionary signatures' of conservation for different functional elements.

4.4.1 Selective Pressures On Different Functional Elements

Different functional elements have different selective pressures (due to their structure and other characteristics); some changes (insertions, deletions, or mutations) that can be extremely harmful to one functional element may be innocuous to another. By figuring out what the "signatures" are for different elements, we can more accurately annotate a region by observing the patterns of conservation it shows.

Importantly, the pattern of conservation has a distinct phylogenetic structure. More similar species (mammals) group together with shared conserved domains that fish lack, suggesting a mammalian specific innovation, perhaps for regulatory elements not shared by fish. Meanwhile, some features are globally conserved, suggesting a universal significance, such as protein coding. Initial approximate annotation of protein coding regions in the human genome was possible using the simple heuristic that if it was conserved from human to fish it likely served as a protein coding region.

An interesting idea for a final project would be to map divergences in the multiple alignment and call these events "births" of new coding elements. By focusing on a particular element (say microRNAs) one could identify periods of innovation and isolate portions of a phylogenetic tree enriched for certain classes of these elements.

[3]See Chapter 7: Hidden Markov Models I

The rest of the chapter will focus on quantifying the degree to which a sequence follows a given pattern. Kellis compared the process of evolution to exploring a fitness landscape, with the fitness score of a particular sequence constrained by the function it encodes. For example, protein coding genes are constrained by selection on the translated product, so synonymous substitutions in the third base pair of a codon are tolerated.

Below is a summary of the expected patterns followed by various functional elements:

- Protein–coding genes exhibit particular frequencies of codon substitution as well as reading frame conservation. This makes sense because the significance of the genes is the proteins they code for; therefore, changes that result in the same or similar amino acids can be easily tolerated, while a tiny change that drastically changes the resulting protein can be considered disastrous. In addition to the error correction of the mismatch repair system and DNA polymerase itself, the redundancy of the genetic code provides an additional level of intrinsic error correction/tolerance.

- Structural RNA is selected based on the secondary sequence of the transcribed RNA, and thus requires compensatory changes. For example, some RNA has a secondary stem–loop structure such that sections of its sequence bind to other sections of its sequence in its "stem", as shown in figure 4.9.

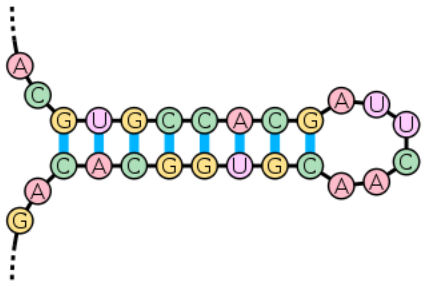


Figure 4.9: RNA with secondary stem–loop structure

  Imagine that a nucleotide (A) and its partner (T) bind to each other in the stem, and then both mutate to a C and G, respectively. Since they are still complementary, this is a compensatory change that maintains its secondary structure. However, if just the T mutated to an A, they would no longer be complementary, so this mutation would not be maintained by evolution, as it ruins the secondary structure. Therefore, in RNA structures, the amount of change to the secondary structure (e.g. stem–loop) is more important than the amount of change in the primary structure (just the sequence). Understanding the effects of changes in RNA structure requires knowledge of the secondary structure. The likely secondary structure of an RNA can be determined by modeling the stability of many possible conformations and choosing the most likely conformation.

- MicroRNA is a molecule that is ejected from the nucleus into the cytoplasm. Their characteristic trait is that they are also have the hairpin (stem–loop) structure illustrated in figure 4.9, but a section of the stem is complementary to a portion of mRNA.

  When microRNA binds its complementary sequence to the respective portion of mRNA, it degrades the mRNA. This means that it is a post–transcriptional regulator, since it's being used to limit the production of a protein (translation) after transcription. MicroRNA is conserved differently than structural RNA. Due to its binding to an mRNA target, the region of binding is much more conserved to maintain target specificity.

- Finally, regulatory motifs are conserved in sequence (to bind particular interacting protein partners) but not necessarily in location. Regulatory motifs can move around since they only need to recruit a factor to a particular region. Small changes (insertions and deletions) that preserve the consensus of the motif are tolerated, as are changes upstream and downstream that move the location of the motif.

When trying to understand the role of conservation in functional class prediction, an important question is how much of observed conservation can be explained by known patterns. Even after accounting for

"random" conservation, roughly 60% of non–random conservation in the fly genome was not accounted for — that is, we couldn't identify it as a protein–coding gene, RNA, microRNA, or regulatory motif. The fact that they remain conserved however suggests a functional role. That so much conserved sequence remains poorly understood underscores that many exciting questions remain to be answered. One final project for 6.047 in the past was using clustering (unsupervised learning) to account for the other conservation. It developed into an M.Eng project, and some clusters were identified, but the function of these clusters was, and is, still unclear. It's an open problem!

4.5 Protein–Coding Signatures

In slide 12, we see three examples of conservation: an intronic sequence with poor conservation, a coding region with high conservation, and a non–coding region with high conservation, meaning it is probably a functional element. The important characteristic of protein–coding regions to remember is that codons (triples of nucleotides) code for amino acids, which make up proteins. This results in the evolutionary signature of protein–coding regions, as shown in slide 13: (i) reading–frame conservation and (ii) codon–substitution patterns. The intuition for this signature is relatively straightforward.

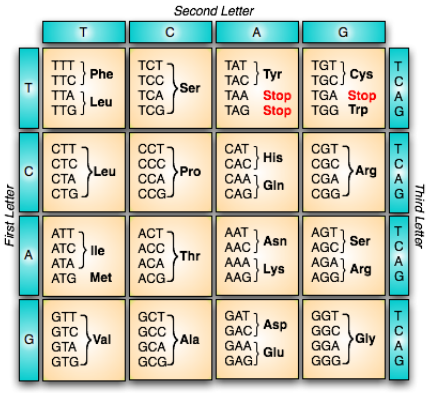


Figure 4.10: Some point mutations to DNA sequence do not change protein translation

Firstly, reading frame conservation makes sense, since an insertion or deletion of one or two nucleotides will "shift" how all the following codons are read. However, if an insertion or deletion happens in a multiple of 3, the other codons will still be read in the same way, so this is a less significant change. Secondly, it makes sense that some mutations are less harmful than others, since different triplets can code for the same amino acids (a conservative substitution, as evident from the matrix below), and even mutations that result in a different amino acid may be evolutionarily neutral if the substitutions occur with similar amino acids in a domain of the protein where exact amino acid properties are not required. These distinctive patterns allow us to "color" the genome and clearly see where the exons are, as shown in slides 14–16.

4.5.1 Reading–Frame Conservation (RFC)

By scoring the pressure to stay in the same reading frame we can quantify how likely a region is to be protein–coding or not. As shown in slide 20, we can do this by having a target sequence (Scer, the genome of *S. cerevisiae*), and then aligning a selecting sequence (Spar, *S. paradoxus*) to it and calculating what proportion of the time the selected sequence matches the target sequence's reading frame. Since we don't know where the reading frame starts in the selected sequence, we align three times ($\text{Spar}_{\text{f1}}, \text{Spar}_{\text{f2}}, \text{Spar}_{\text{f3}}$) to try all possible offsets, and then choose the alignment where the selected sequence is most often in sync with the target sequence. Finally, for the best alignment, we calculate the percentage of nucleotides that are out of frame — if it is above a cutoff, this selected species "votes" that this region is a protein–coding region , and if it is low, this species "votes" that this is an intergenic region. The "votes" are tallied from all the species to sum to the RFC score.

Figure 4.11: Two alignments showing conservation pattern differences between gene and intergenic sequences. Red boxes represent gaps that shift the coding frame, and gray boxes are non-frame-shifting gaps (in multiples of three). Green regions are conserved, and yellow ones are mutated. Note the pattern of "match, match, mismatch" in the protein-coding sequence that indicates synonymous mutations.

This method is not robust to sequencing error. We can compensate for these errors by using a smaller scanning window and observing local reading frame conservation.

The method was shown to have 99.9% specificity and 99% sensitivity when applied to the yeast genome. When applied to 2000 hypothetical ORFs (open reading frames, or proposed genes)[4] in yeast, it rejected 500 of these putative protein coding genes as not being protein coding.

Similarly, 4000 hypothetical genes in the human genome were rejected by this method. This model created a specific hypothesis (that these DNA sequences were unlikely to code for proteins) that has subsequently been supported with experimental confirmation that the regions do not code for proteins in vivo.[5]

Figure 4.12: Red boxes represent frame-shifting gaps, and gaps in multiples of three are uncolored. Conserved and mutated regions are green and yellow, respectively.

This represents an important step forward for genome annotation, because previously it was difficult to conclude that a DNA sequence was non–coding simply from lack of evidence. By narrowing the focus and creating a new null hypothesis (that the gene in question appears to be a non–coding gene) it became much

[4]Kellis M, Patterson N, Endrizzi M, Birren B, Lander E. S. 2003. Sequencing and comparison of yeast species to identify genes and regulatory elements. Science. 423: 241–254.

[5]Clamp M et al. 2007. Distinguishing protein–coding and noncoding genes in the human genome. PNAS. 104: 19428–19433.

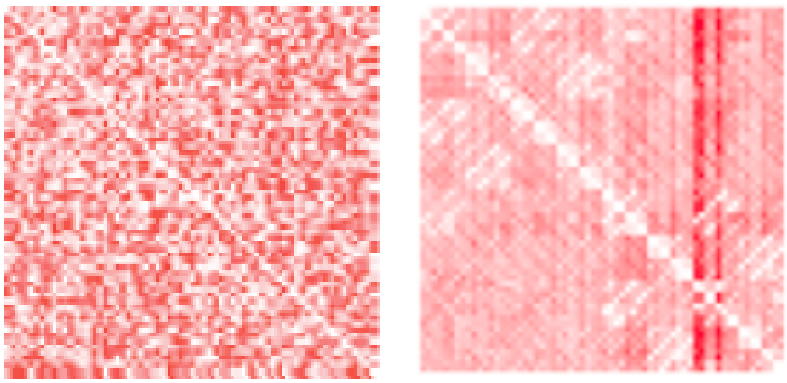
easier to not only accept coding genes, but to reject non–coding genes with computational support. During the discussion of reading frame conservation in class, we identified an exciting idea for a final project which would be to look for the birth of new functional proteins resulting from frame shift mutations.

4.5.2 Codon–Substitution Frequencies (CSFs)

The second signature of protein coding regions, the codon substitution frequencies, acts on multiple levels of conservation. To explore these frequencies, it is helpful to remember that codon evolution can be modeled by conditional probability distributions (CPDs) — the likelihood of a descendant having a codon b where an ancestor had codon a an amount of time t ago.

The most conservative event is exact maintenance of the codon. A mutation that codes for the same amino acid may be conservative but not totally synonymous, because of species specific codon usage biases. Even mutations that alter the identity of the amino acid might be conservative if they code for amino acids with similar biochemical properties. We use a CPD in order to capture the net effect of all of these considerations. To calculate these CPDs, we need a "rate" matrix, Q, which measures the exchange rate for a unit time; that is, it indicates how often codon a in species 1 is substituted for codon b in species 2, for a unit branch length. Then, by using $e^{Q}t$, we can estimate the frequency of substitution at time t.

The intuition, as shown in slide 31, is that as time increases, the probability of substitutions increase, while at the "initial" time ($t = 0$), $e^{Q}t$ is the identity matrix, since every codon is guaranteed to be itself. But how do we get the rate matrix? Q is "learned" from the sequences, by using Expectation–Maximization, for example. Given the parameters of the model, we can use Felsenstein's algorithm[1] to compute the probability of any alignment, while taking into account phylogeny, given the substitution model (the E–step). Then, given the alignments and phylogeny, we can choose the parameters (the rate matrix: Q, and branch lengths: t) that maximize the likelihood of those alignments in the M–step; for example, to estimate Q, we can count the number of times one codon is substituted for another in the alignment.



(a) Rate matrix Q_N estimated from non–coding regions

(b) Rate matrix Q_c estimated from known coding regions.

Figure 4.13: Rate matrices for the null and alternate models.

Now that we know how to obtain our model, we note that, given the specific pattern of codon substitution frequencies for protein–coding, we want two models so that we can distinguish between coding and non–coding regions. The images above show the two distinct rate matrices, one each for genes and intergenic regions, where a lighter color means the substitution is more likely. A number of salient features present themselves in the codon substitution matrix (CSM) for genes. Note that the main diagonal element has been removed, because the frequency of a triplet being exchanged for itself will obviously be much higher than any other exchange. Nevertheless,

1. it is immediately obvious that there is a strong diagonal element in the protein coding regions.
2. We also note certain high–scoring off diagonal elements in the coding CSM: these are substitutions that are close in function rather than in sequence, such as 6–fold degenerate codons or very similar

amino acids.

3. We also note dark vertical stripes, which indicate these substitutions are especially unlikely. These columns correspond to stop codons, since substitutions to this triplet would significantly alter protein function, and thus are strongly selected against.

On the other hand, in the matrix for intergenic regions, the exchange rates are more uniform. In these regions, what matters is the *mutational proximity*, i.e. the edit distance or number of changes from one sequence to another. Genetic regions are dictated by *selective proximity*, or the similarity in amino acid sequence of the protein resulting from the gene.

Now that we have the two rate matrices for the two regions, we can calculate the probabilities that each matrix generated the genomes of the two species. This can be done by using Felsenstein's algorithm, and adding up the "score" for each pair of corresponding codons in the two species. Finally, we can calculate the likelihood ratio that the alignment came from a coding region to a non–coding region by dividing the two scores — this demonstrates our confidence in our annotation of the sequence. If the ratio is greater than 1, we can guess that it is a coding region, and if it is less than 1, then it is a non–coding region. For example, in Figure 4.11, we are very confident about the respective classifications of each region.

It should be noted, however, that although the "coloring" of the sequences confirms our classifications, the likelihood ratios are calculated independently of the 'coloring,' which uses our knowledge of synonymous or conservative substitutions. This further implies that this method automatically infers the genetic code from the pattern of substitutions that occurs, simply by looking at the high scoring substitutions. In species with a different genetic code, the patterns of codon exchange will be different; for example, in Candida albumin, the CTG codes for serine (polar) rather than leucine (hydrophobic), and this can be deduced from the CSMs. However, no knowledge of this is required by the method; instead, we can deduce this *a posteriori* from the CSM.

In summary, we are able to distinguish between non–coding and coding regions of the genome based on their evolutionary signatures, by creating two separate 64 by 64 *rate matrices*: one measuring the rate of codon substitutions in coding regions, and the other in non–coding regions. The rate matrix gives the exchange rate of codons or nucleotides over a unit time.

We used the two matrices to calculate two probabilities for any given alignment: the likelihood that it came from a coding region and the likelihood that it came from a non–coding region. Taking the *likelihood ratio* of these two probabilities gives a measure of confidence that the alignment is protein–coding. Using this method we can pick out regions of the genome that evolve according to the protein coding signature.

We will see later how to combine this likelihood ratio approach with phylogenetic methods to find evolutionary patterns of protein coding regions.

However, this method only lets us find regions that are selected at the translational level. The key point is that here we are measuring for only protein coding selection. We will see today how we can look for other conserved functional elements that exhibit their own unique signatures.

4.5.3 Classification of *Drosophila* Genome Sequences

We have seen that using these RFC and CSF metrics allows us to classify exons and introns with extremely high specificity and sensitivity. The classifiers that use these measures to classify sequences can be implemented using a HMM or semi–Markov conditional random field (SMCRF). CRFs allow the integration of diverse features that do not necessarily have a probabilistic nature, whereas HMMs require us to model everything as transition and emission probabilities. CRFs will be discussed in an upcoming lecture. One might wonder why these more complex methods need to be implemented, when the simpler method of checking for conservation of the reading frame worked well. The reason is that in very short regions, insertions and deletions will be very infrequent, even by chance, so there won't be enough signal to make the distinction between protein–coding and non–protein–coding regions. In the figure below, we see a DNA sequence along the x–axis, with the rows representing an annotated gene, amount of conservation, amount of protein–coding evolutionary signature, and the result of Viterbi decoding using the SMCRF, respectively.

This is one example of how utilization of the protein–coding signature to classify regions has proven very successful. Identification of regions that had been thought to be genes but that did not have high protein–coding signatures allowed us to strongly reject 414 genes in the fly genome previously classified as CGid–only

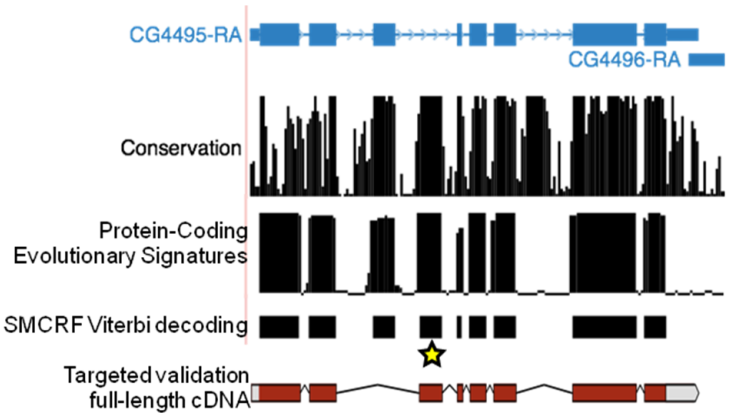


Figure 4.14: Evolutionary signatures can predict new genes and exons. The star denotes a new exon, which was predicted using the three comparative genomics tests, and later verified using cDNA sequencing.

genes, which led FlyBase curators to delete 222 of them and flag another 73 as uncertain. In addition, there were also definite false negatives, as functional evidence existed for the genes under examination. Finally, in the data, we also see regions with both conversation, as well as a large protein–coding signature, but had not been previously marked as being parts of genes, as in the figure above. Some of these have been experimentally tested and have been show to be parts of new genes or extensions of existing genes. This underscores the utility of computational biology to leverage and direct experimental work.

4.5.4 Leaky Stop Codons

Stop codons (TAA, TAG, TGA in DNA and UAG, UAA, UGA in RNA) typically signal the end of a gene. They clearly reflect translation termination when found in mRNA and release the amino acid chain from the ribosome. However, in some unusual cases, translation is observed beyond the first stop codon. In instances of single read–through, there is a stop codon found within a region with a clear protein–coding signature followed by a second stop–codon a short distance away. An example of this in the human genome is given in Figure 4.15. This suggests that translation continues through the first stop codon. Instances of double read–through, where two stop codons lie within a protein coding region, have also been observed. In these instances of stop codon suppression, the stop codon is found to be highly conserved, suggesting that these skipped stop codons play an important biological role.

Translational read–through is conserved in both flies, which have 350 identified proteins exhibiting stop codon read–through, and humans, which have 4 identified instances of such proteins. They are observed mostly in neuronal proteins in adult brains and brain expressed proteins in Drosophila.

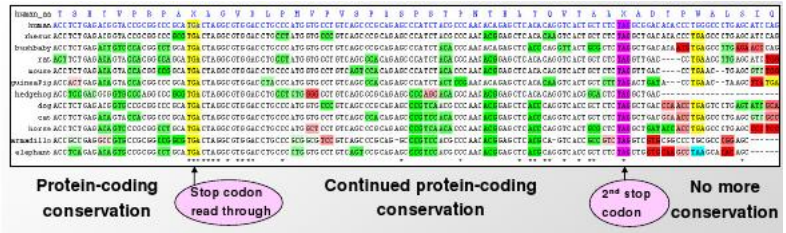


Figure 4.15: OPRL1 neurotransmitter: one of the four novel translational read–through candidates in the human genome. Note that the region after the first stop codon exhibits an evolutionary signature similar to that of the coding region before the stop codon, indicating that the stop codon is "suppressed".

The kelch gene exhibits another example of stop codon suppression at work. The gene encodes two ORFs with a single UGA stop codon between them. Two proteins are translated from this sequence, one from the first ORF and one from the entire sequence. The ratio of the two proteins is regulated in a tissue–specific

manner. In the case of the kelch gene, a mutation of the stop codon from UGA to UAA results in a loss of function, suggesting that tRNA suppression is the mechanism behind stop codon suppression.

An additional example of stop codon suppression is, Caki, a protein active in the regulation of neurotransmitter release in *Drosophila*. Open reading frames (ORFs) are DNA sequences which contain a start and stop codon. In Caki, reading the gene in the first reading frame (Frame 0) results in significantly more ORFs than reading in Frame 1 or Frame 2 (a 440 ORF excess). Figure 4.16 lists twelve possible interpretations for the ORF excess. However, because the excess is observed only in Frame 0, only the first 4 interpretations are likely:

- Stop–codon readthrough: the stop codon is suppressed when the ribosome pulls in tRNA that pairs incorrectly with the stop codon.
- Recent nonsense: Perhaps some recent nonsense mutation is causing stop codon readthrough.
- A to I editing: Unlike we previously thought, RNA can still be edited after transcription. In some case the A base is changed to an I, which can be read as a G. This could change a TGA stop codon to a TGG, which encodes an amino acid. However, this phenomenon is only found in a couple of cases.
- Selenocysteine, the "21st amino acid": Sometimes when the TGA codon is read by a certain loop which leads to a specific fold of the RNA, it can be decoded as selenocysteine. However, this only happens in four fly proteins, so can't explain all of stop codon suppression.

Among these four, three of them (recent nonsense, A to I editing, and selenocysteine) account for only 17 of the cases. Hence, it seems that read–through must be responsible for all if not most of the remaining cases. In addition, biased stop codon usage is observed hence ruling out other processes such as alternative splicing (where RNA exons following transcription are reconnected in multiple ways leading to multiple proteins) or independent ORFs.

Interpretation	Frame 0	Frame 1	Frame 2
Readthrough	√		
Recent nonsense	√		
A -> I editing	√	Only 17 cases.	
Selenocysteine	√		
Alternative splicing	√	√	√
Dicistronic	√	√	√
Cryptic promotor	√	√	√
Hopping	√	√	√
Antisense	√	√	√
Chance	√	√	√
Frame shift		√	√
CDS overlaps stop		√	√
# with PCSF > 0	662	219	225

Figure 4.16: Various interpretations of stop codon suppression. See text for explanation.

Read–through regions can be determined in a single species based on their pattern of codon usage. The Z–curve as shown in Figure 4.17 measures codon usage patterns in a region of DNA. From the figure, one can observe that the read–through region matches the distribution before the regular stop codon. After the second stop however, the region matches regions found after regular stops.

Another suggestion offered in class was the possibility of ribosome slippage, where the ribosome skips some bases during translation. This might cause the ribosome to skip past a stop codon. This event occurs in bacterial and viral genomes, which have a greater pressure to keep their genomes small, and therefore can

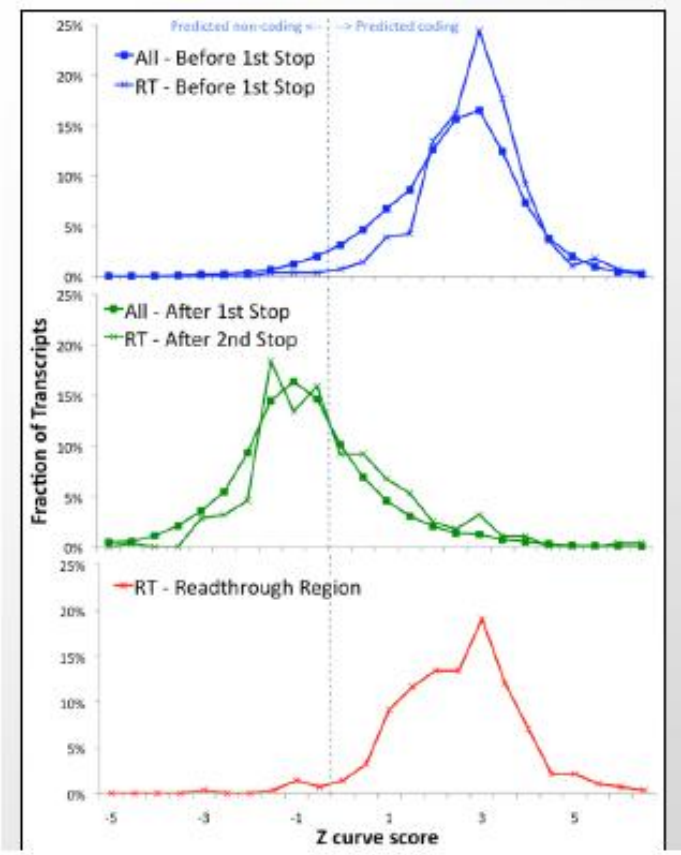


Figure 4.17: Z–curve for *Caki*. Note that the codon usage in the read through region is similar to that in the region before the first stop codon.

use this slipping technique to read a single transcript in each different reading frame. However, humans and flies are not under such extreme pressure to keep their genomes small. Additionally, we showed above that the excess we observe beyond the stop codon is frame specific to frame 0, suggesting that ribosome slipping is not responsible.

Cells are stochastic in general and most processes tolerate mistakes at low frequencies. The system isn't perfect and stop codon leaks happen. However, the following evidence suggests that stop codon read–through is not random but instead subject to regulatory control:

- Perfect conservation of read–through stop codons is observed in 93% of cases, which is much higher than the 24% found in background.
- Increased conservation is observed upstream of the read–through stop codon.
- Stop codon bias is observed. TGAC is the most frequent sequence found at the stop codon in read–through and the least frequent found at normal terminated stop codons. It is known to be a "leaky" stop codon. TAAA is found almost universally only in non–read–through instances.
- Unusually high numbers of GCA repeats observed through read–through stop Conn.
- Increased RNA secondary structure is observed following transcription suggesting evolutionarily conserved hairpins.

4.6 microRNA (miRNA) genes

One example of functional genomic regions subject to high levels of conservation are sequences encoding microRNAs (miRNAs). miRNAs are RNA molecules that bind to complementary sequences in the 3' untranslated region of targeted mRNA molecules, causing gene silencing.

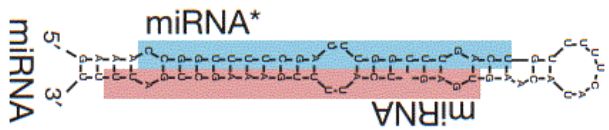


Figure 4.18: The hairpin structure of a microRNA. Note that miRNA* denotes the strand on the opposite side of the hairpin, which has the same sequence as the mRNA molecules that are suppressed by the miRNA.

How do we find evolutionary signatures for miRNA genes and their targets, and can we use these to gain new insights on their biological functions? We will see that this is a challenging task, as miRNAs leave a highly conserved but very subtle evolutionary signal.

4.6.1 Computational Challenge

Predicting the location of miRNA genes and their targets is a computationally challenging problem. We can look for "hairpin" regions, where we find nucleotide sequences that are complementary to each other and predict a hairpin structure. But out of 760,355 miRNA–like hairpins found in the cell, only 60–100 were true miRNAs. So to make any test that will give us regions statistically likely to be miRNAs, we need a test with 99.99% specificity.

Figure 4.19 is an example of the conservation pattern for miRNA genes. You can see the two hairpin structures conserved in the red and blue regions, with a region of low conservation in the middle. This pattern is characteristic of miRNAs.

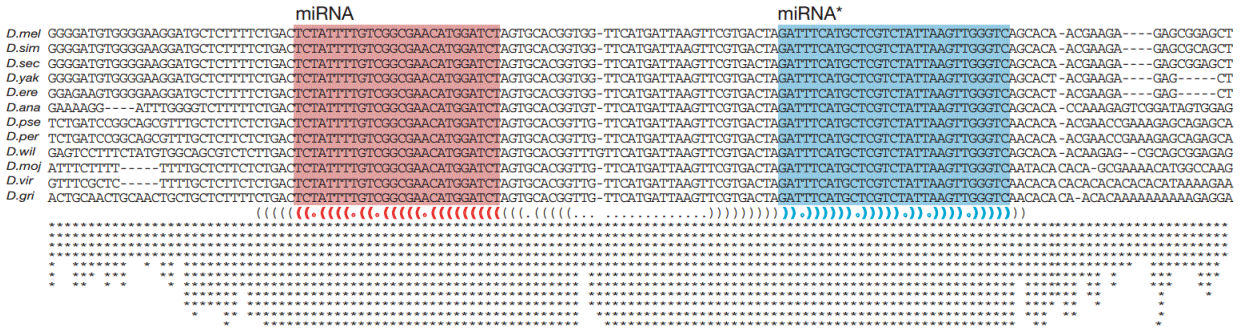


Figure 4.19: Characteristic conservation pattern of miRNAs. The number of asterisks below a nucleotide indicates the number of species where it is conserved. The blue and red highly conserved regions represent the complementary strands of the miRNA, as in figure 4.18.

By analyzing evolutionary and structural features specific to miRNA, we can use combinations of these features to pick out regions of miRNAs with >4,500 enrichment compared to random hairpins. The following are examples of features that help pick out miRNAs:

- miRNAs bind to highly conserved target motifs in the 3' UTR
- miRNAs can be found in introns of known genes
- miRNAs have a preference for the positive strand of DNA and for transcription factors
- miRNAs are not found in exonic and repetitive elements of the genome

We can combine several features into one test by using a decision tree, as illustrated in figure 4.20. At each node of the tree, a test is applied which determines which branch will be followed next. The tree is traversed starting from the root until a terminal node is reached, at which point the tree will output a classification. A decision tree can be trained using a body of classified genome subsequences, after which it can be used to predict whether new subsequences are miRNAs or not. In addition, many decision trees can be combined into a "random forest," where several decision trees are trained. When a new nucleotide sequence needs to be classified, each tree votes on whether or not it is an miRNA, and then the votes are aggregated to determine the final classification.

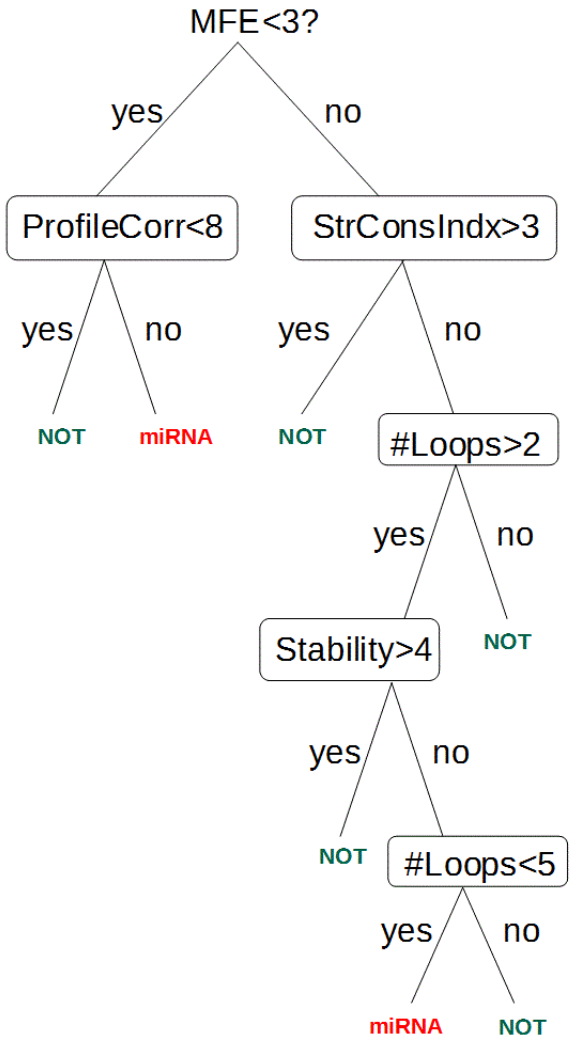


Figure 4.20: A possible decision tree for miRNA detection. The features used in this tree are minimum free energy, conservation profile correlation, structure conservation index, number of loops, and stability.

Applying this technique to the fly genome showed 101 hairpins above the 0.95 cutoff, rediscovering 60 of 74 of known miRNAs, predicting 24 novel miRNAs that were experimentally validated, and finding an additional 17 candidates that showed evidence of diverse function.

4.6.2 Unusual miRNA Genes

The following four "surprises" were found when looking at specific miRNA genes:

Surprise 1 Both strands might be expressed and functional. For instance, in the miR–iab–4 gene, expression of

the sense and antisense strands are seen in distinct embryonic domains. Both strands score > 0.95 for miRNA prediction.

Surprise 2 Some miRNAs might have multiple 5' ends for a single miRNA arm, giving evidence for an imprecise start site. This could give rise to multiple mature products, each potentially with its own functional targets.

Surprise 3 High scoring miRNA* regions (the star arm is complementary to the actual miRNA sequence) are very highly expressed, giving rise to regions of the genome that are both highly expressed and contain functional elements.

Surprise 4 Both miR–10 and miR–10* have been shown to be very important Hox regulators, leading to the prediction that miRNAs could be "master Hox regulators". Pages 10 and 11 of the first set of lecture 5 slides show the importance of miRNAs that form a network of regulation for different Hox genes.

4.7 Regulatory Motifs

Another class of functional element that is highly conserved across many genomes contains *regulatory motifs*. A regulatory motif is a highly conserved sequence of nucleotides that occurs many times throughout the genome and serves some regulatory function. For instance, these motifs might characterize enhancers, promoters, or other genomic elements.

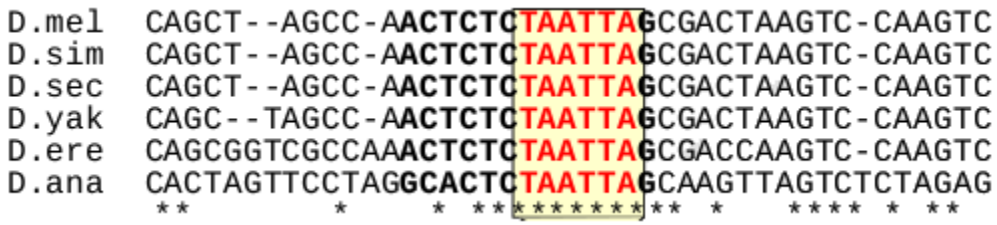


Figure 4.21: TAATTA is a hexamer that appears as a conserved element throughout the genome in many different functional elements, including here. It is an example of a regulatory motif.

4.7.1 Computationally Detecting Regulatory Motifs

Computational methods have been developed to measure conservation of regulatory motifs across the genome, and to find new unannotated motifs *de novo*. Known motifs are often found in regions with high conservation, so we can increase our testing power by testing for conservation, and then finding signatures for regulatory motifs.

Evaluating the pattern of conservation for known motifs versus the "null model" of regions without motifs gives the following signature:

Conservation within:	Gal4 (known motif region)	Controls
All intergenic regions	13%	2%
Intergenic: coding	13%: 3%	2%:7%
Upstream: downstream	12: 0	1:1

So as we can see, regions with regulatory motifs show a much higher degree of conservation in intergenic regions and upstream of the gene of interest.

To discover novel motifs, we can use the following pipeline:

- Pick a motif "seed" consisting of two groups of three non–degenerate characters with a variable size gap in the middle.

- Use a conservation ratio to rank the seed motifs
- Expand the seed motifs to fill in the bases around the seeds using a hill climbing algorithm.
- Cluster to remove redundancy.

Discovering motifs and performing clustering has led to the discovery of many motif classes, such as tissue specific motifs, function specific motifs, and modules of cooperating motifs.

4.7.2 Individual Instances of Regulatory Motifs

To look for expected motif regions, we can first calculate a *branch–length score* for a region suspected to be a regulatory motif, and then use this score to give us a confidence level of how likely something is to be a real motif.

The branch length score (BLS) sums evidence for a given motif over branches of a phylogenetic tree. Given the pattern of presence or absence of a motif in each species in the tree, this score evaluates the total branch length of the sub–tree connecting the species that contain the motif. If all species have the motif, the BLS is 100%. Note more distantly related species are given higher scores, since they span a longer evolutionary distance. If a predicted motif has spanned such a long evolutionary time frame, it is likely it is a functional element rather than just a region conserved by random chance.

To create a null model, we can choose control motifs. The null model motifs should be chosen to have the same composition as the original motif, to not be too similar to each other, and to be dissimilar from known motifs. We can get a confidence score by comparing the fraction of motif instances to control motifs at a given BLS score.

4.8 Current Research Directions

4.9 Further Reading

4.10 Tools and Techniques

4.11 What Have We Learned?

Bibliography

[1] Joseph Felsenstein. Evolutionary trees from dna sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981. 10.1007/BF01734359.

CHAPTER

FIVE

GENOME ASSEMBLY AND WHOLE-GENOME ALIGNMENT

Melissa Gymrek, Liz Tsai, Rebecca Taft (2012), Keshav Dhandhania (2012)

Figures

5.1 Introduction

In the previous chapter, we saw the importance of comparative genomics analysis for discovering functional elements. In "part IV" of this book, we will see how we can use comparative genomics for studying gene evolution across species and individuals. In both cases however, we assumed that we had access to complete and aligned genomes across multiple species.

In this chapter, we will study the challenges of genome assembly and whole-genome alignment that are the foundations of whole-genome comparative genomics methodologies. First, we will study the core algorithmic principles underlying many of the most popular genome assembly methods available today. Second, we will study the problem of whole-genome alignment, which requires understanding mechanisms of genome rearrangement, segmental duplication, and other translocations. The two problems of genome assembly and whole-genome alignment are similar in nature, and we close by discussing some of the parallels between them.

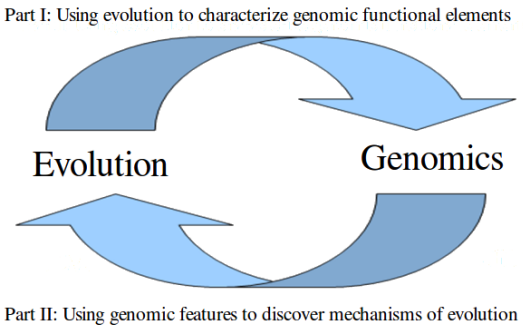


Figure 5.1: We can use evolutionary signatures to find genomic functional elements, and in turn can study mechanisms of evolution by looking at genomic patterns.

5.2 Genome Assembly I: Overlap-Layout-Consensus Approach

Many areas of research in computational biology rely on the availability of complete whole-genome sequence data. Yet the process to sequence a whole-genome is itself non-trivial and an area of active research. The problem lies in the fact that current genome-sequencing technologies cannot continuously read from one end of a long genome sequence to the other; they can only accurately sequence small sections of at most 1000-2000 base pairs, called *reads*. Therefore, in order to construct a sequence of millions or billions of base pairs (such as the human genome), computational biologists must find ways to combine smaller reads into larger, continuous DNA sequences.

This section will examine one of the most successful early methods for computationally assembling a genome from a set of DNA reads, called shotgun sequencing (Figure 5.2). Shotgun sequencing involves randomly shearing multiple copies of the same genome into many small fragments, as if the DNA were shot with a shotgun. Typically, the DNA is actually fragmented using either sonication (brief bursts from an ultrasound) or a targeted enzyme designed to cleave the genome at specific sequence motifs. Both of these methods can be tuned to create fragments of varying sizes.

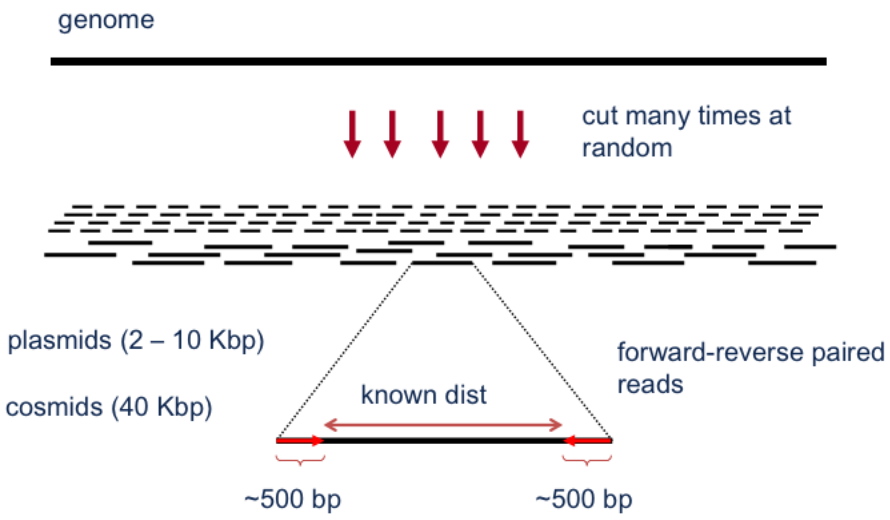


Figure 5.2: Shotgun sequencing involves randomly shearing a genome into small fragments so they can be sequenced, and then computationally reassembling them into a continuous sequence

After the DNA has been amplified and fragmented, the technique developed by Frederick Sanger in 1977 called chain-termination sequencing is used to sequence the fragments. A detailed description of this method is beyond the scope of this book, but the result is many sequences of bases with corresponding per-base quality scores, indicating the probability that each base was called correctly. The shorter fragments can be fully sequenced, but the longer fragments can only be sequenced at each of their ends since the quality diminishes significantly after about 500-900 base pairs. These paired-end reads are called *mate pairs*. In the rest of this section, we discuss how to use the reads to construct much longer sequences, up to the size of entire chromosomes.

5.2.1 Finding overlapping reads

To combine the DNA fragments into larger segments, we must find places where two or more reads overlap, i.e. where the beginning sequence of one fragment matches the end sequence of another fragment. For example, given two fragments such as ACGTTGACCGCATTCGCCATA and GACCGCATTCGCCATACGGCATT, we can construct a larger sequence based on the overlap: ACGTTGACCGCATTCGCCATACGGCATT (Figure 5.3).

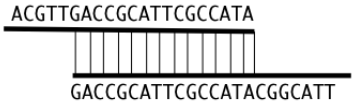


Figure 5.3: Given two shorter fragments with overlapping sequences, we can construct one longer sequence

One method for finding matching sequences is the Needleman-Wunsch dynamic programming algorithm, which was discussed in chapter 2. The Needleman-Wunsch method is impractical for genome assembly, however, since we would need to perform millions of pairwise-alignments, each taking $O(n^2)$ time, in order to construct an entire genome from the DNA fragments.

A better approach is to use the BLAST algorithm (discussed in chapter 3) to hash all the k-mers (unique sequences of length k) in the reads and find all the locations where two or more reads have one of the k-mers in common. k can be any number smaller than the size of the reads, but varies depending on the desired sensitivity and specificity. One popular overlap-layout-consensus assembler called Arachne uses $k = 24$ [2].

Given the matching k-mers, we can align each of the corresponding reads and discard any matches that are less than 97% similar. We do not require that the reads be identical since we allow for the possibility of sequencing errors and polymorphism in the genome (e.g., humans have two copies of every gene which may or may not be the same).

5.2.2 Merging reads into contigs

Using the techniques described above to find overlaps between DNA fragments, we can piece together larger segments of continuous sequences called *contigs*. One way to visualize this process is to create a graph in which all the nodes represent reads, and the edges represent overlaps between the reads (Figure 5.4). By removing the transitively inferable overlaps, we can create a chain of reads that have been ordered to form a larger contig.

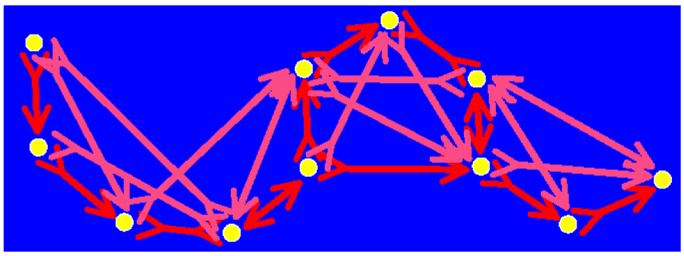


Figure 5.4: We can visualize the process of merging fragments into contigs by letting the nodes in a graph represent reads and edges represent overlaps. By removing the transitively inferable edges (the pink edges in this image), we are left with chains of reads ordered to form contigs

In theory, we should be able to use the above approach to create large contigs from our reads as long as we have adequate coverage of the given region. In practice, we often encounter large sections of the genome that are extremely repetitive and as a result are difficult to assemble. For example, it is unclear exactly how to align the following two sequences: ATATATAT and ATATATATAT. Due to the extremely low variation in the sequence pattern, they could overlap in any number of ways. Furthermore, these repetitive regions may appear in multiple locations in the genome, and it is difficult to determine which reads come from which locations. Contigs made up of these ambiguous, repetitive reads are called *overcollapsed contigs*.

In order to determine which sections are overcollapsed, it is often possible to quantify the depth of coverage of fragments making up each contig. If one contig has significantly more coverage than the others, it is a likely candidate for an overcollapsed region. Additionally, several unique contigs may overlap one contig in the same location, which is another indication that the contig may be overcollapsed (Figure 5.5).

After fragments have been assembled into contigs up to the point of a possible repeated section, the result is a graph in which the nodes are contigs, and the edges are links between unique contigs and overcollapsed contigs (Figure 5.6).

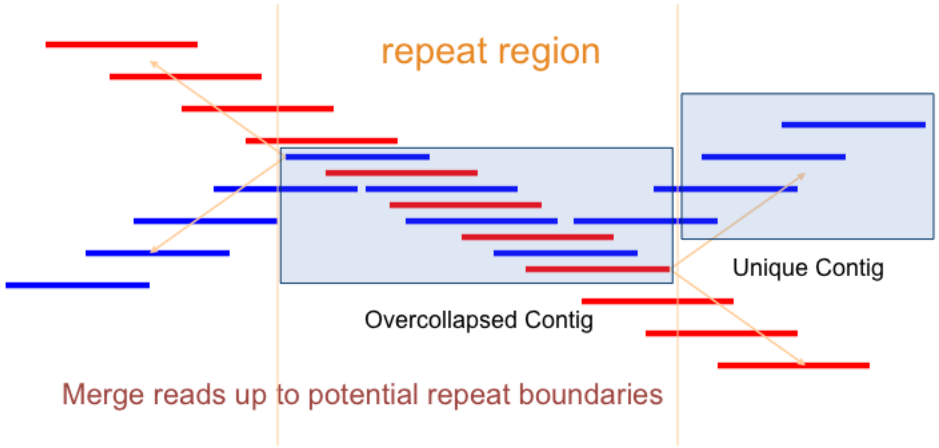


Figure 5.5: Overcollapsed contigs are caused by repetetive regions of the genome which cannot be distinguished from one another during sequencing. Branching patterns of alignment that arise during the process of merging fragments into contigs are a strong indication that one of the regions may be overcollapsed.

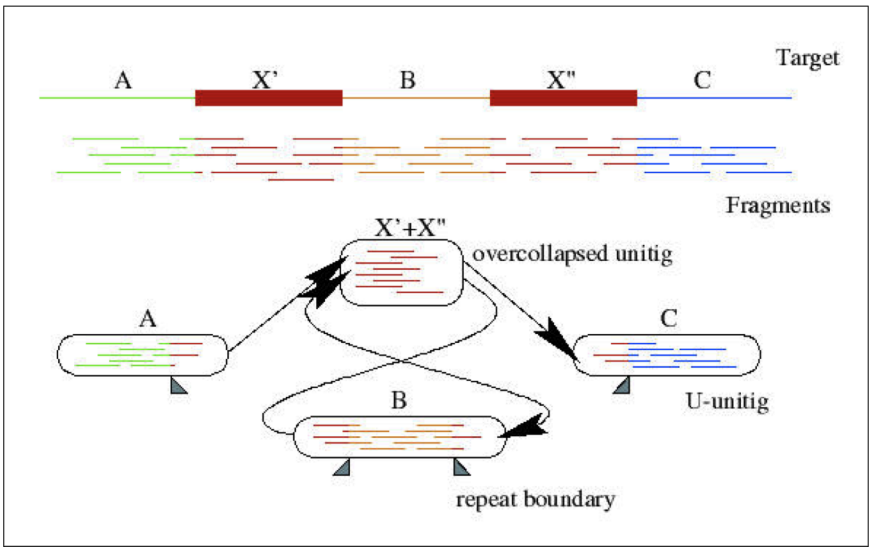


Figure 5.6: In this graph connecting contigs, region X has indegree and outdegree equal to 2. The target seqence shown at the top can be inferred from the links in the graph

5.2.3 Laying out contig graph into scaffolds

Once our fragments are assembled into contigs and contig graphs, we can use the larger mate pairs to link contigs into *supercontigs* or *scaffolds*. Mate pairs are useful both to orient the contigs and to place them in the correct order. If the mate pairs are long enough, they can often span repetitive regions and help resolve the ambiguities described in the previous section (Figure 5.7).

Unlike contigs, supercontigs may contain some gaps in the sequence due to the fact that the mate pairs connecting the contigs are only sequenced at the ends. Since we generally know how long a given mate pair is we can estimate how many base pairs are missing, but due to the randomness of the cuts in shotgun sequencing, we may not have the data available to fill in the exact sequence. Filling in every single gap can

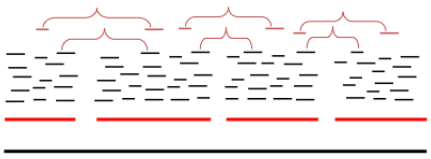


Figure 5.7: Mate pairs are used to link contigs into supercontigs

be extremely expensive, so even the most completely assembled genomes usually contain some gaps.

5.2.4 Deriving consensus sequence

The goal of genome assembly is to create one continuous sequence, so after the reads have been aligned into contigs, we need to resolve any differences between them. As mentioned above, some of the overlapping reads may not be identical due to sequencing errors or polymorphism. We can often determine when there has been a sequencing error when one base disagrees with all the other bases aligned to it. Taking into account the quality scores on each of the bases, we can usually resolve these conflicts fairly easily. This method of conflict resolution is called weighted voting (Figure 5.8). Another alternative is to ignore the frequencies of each base and take the maximum quality letter as the consensus.

Figure 5.8: We derive the multiple alignment consensus sequence by weighted voting at each base

In some cases, it is not possible to derive a consensus if, for example, the genome is heterozygous and there are equal numbers of two different bases at one location. In this case, the assembler must choose a representative.

> ***Did You Know?***
> Since polymorphism can significantly complicate the assembly of diploid genomes, some researchers induce several generations of inbreeding in the selected species to reduce the amount of heterozygosity before attempting to sequence the genome.

5.2.5 Dealing with sequencing errors

The Overlap-Layout-Consensus method for assembling a genome as described in this section has proven to be a very successful approach to assembling large genomes with minimal errors. Challenges remain, however, and even the most complete genomes contain some errors and gaps. By tuning the sequencing technology to create as few errors as possible and correcting the remaining errors using weighted voting, we can minimize the number of errors in the resulting whole-genome sequence. By adjusting the read length to span the repetitive regions of the genome, we can correctly resolve these regions and come very close to the ideal of a complete, continuous genome.

In this section, we saw an algorithm to do genome assembly given reads. However, this algorithm works well when the reads are 500 - 900 bases long or more, which is what we get when one does Sanger sequencing. Alternate genome assembly algorithms are required is the reads we get from our sequencing methods are much shorter.

5.3 Genome Assembly II: String graph methods

Shutgun sequencing, which is a more modern and economic method of sequencing, gives reads that around 100 bases in length. The shorler length of the reads results in a lot more repeats of length greater than that of the reads. Hence, we need new and more sophisticated algorithm to do genome assembly correctly.

5.3.1 String graph definition and construction

Before going into the method of string graph based genome assembly, it is convenient to note that the string graph is built with the objective that finding a feasible flow in the string graph and then traversing the edges of the flow should give us the entire genome.

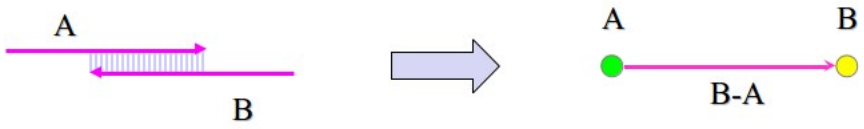


Figure 5.9: Constructing a string graph

Starting from the reads we get from Shotgun sequencing, a string graph is constructed by adding an edge for every pair of overlapping reads. Note that the vertices of the graph denote junctions, and the edges correspond to the string of bases. A single node corresponds to each read, and reaching that node while traversing the graph is equivalent to reading all the bases upto the end of the read corresponding to the node. For example, in figure 5.9, we have two overlapping reads A and B and they are the only reads we have. The corresponding string graph has two nodes and two edges. One edge doesn't have a vextex at its tail end, and has A at its head end. This edge denotes all the bases in read A. The second edge goes from node A to node B, and only denotes the bases in B-A (the part of read B which is not overlapping with A). This way, when we traverse the edges once, we read the entire region exactly once. In particular, notice that we do not traverse the overlap of read A and read B twice.

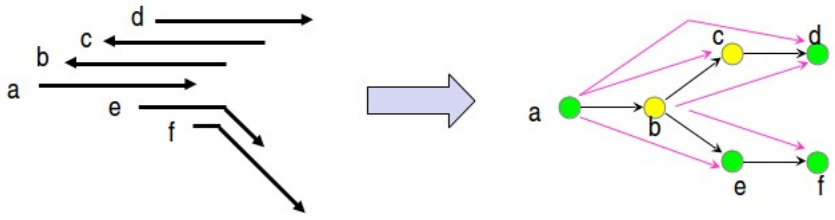


Figure 5.10: Constructing a string graph

There are a couple of subtleties in the string graph (figure 5.10) which need mentioning:-

- We have two different colors for nodes since the DNA can be read in two directions. If the overlap is between the reads as is, then the nodes receive same colors. And if the overlap is between a read and the complementary bases of the other read, then they receive different colors.
- Secondly, if A and B overlap, then there is ambiguity in whether we draw an edge from A to B, or from B to A. Such ambuigity needs to be resolved in a consistent manner at junctions caused due to repeats.

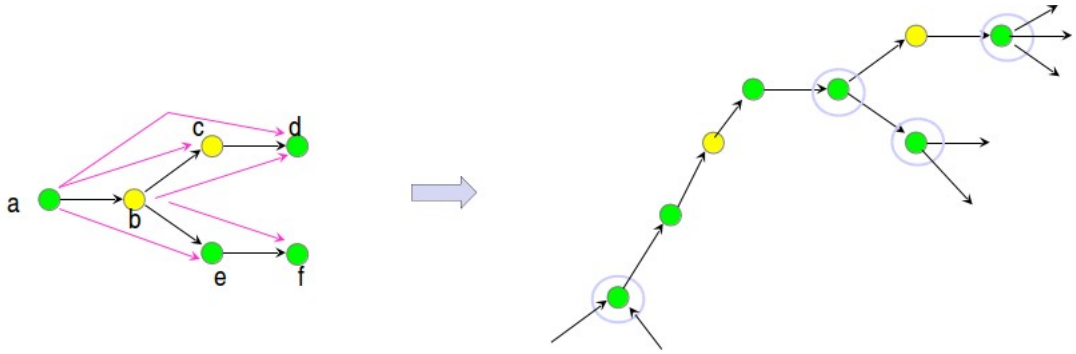


Figure 5.11: Example of string graph undergoing removal of transitive egdes

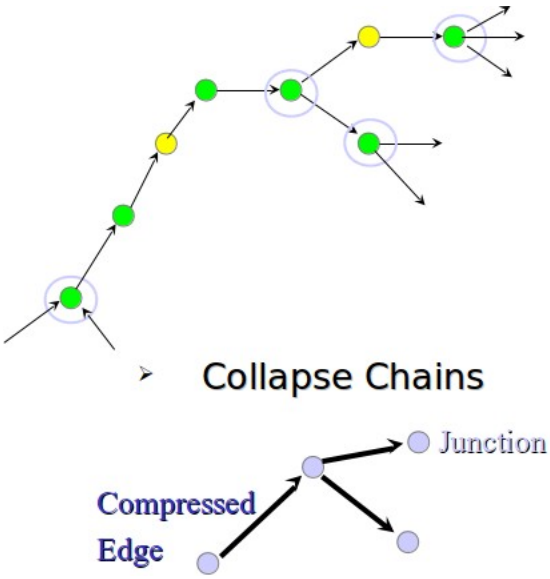


Figure 5.12: Example of string graph undergoing chain collapsing

After constructing the string graph from overlapping reads, we:-

- *Remove transitive edges:* Transitive edges are caused by transitive overlaps, i.e. A overlap B overlaps C in such a way that A overlaps C. There are randomized algorithms which remove transitive edges in O(E) expected runtime. In figure 5.11, you can see the an example of removing transitive edges.

- *Collapse chains:* After removing the transitive edges, the graph we build will have many chains where each node has one incoming edge and one outgoing edge. We collapse all these chains to a single edge. An example of this is shown in figure 5.12.

5.3.2 Flows and graph consistency

After doing everything mentioned above we will get a pretty complex graph, i.e. it will still have a number of junctions due to relatively long repeats in the genome compared to the length of the reads. We will now see how the concepts of flows can be used to deal with repeats.

First, we estimate the weight of each edge by the number of reads we get corresponds to the edge. If we have double the number of reads for some edge than the number of DNAs we sequenced, then it is fair to assume that this region of the genome gets repeated. However, this technique by itself is not accurate enough. Hence sometimes we may make estimates by saying that the weight of some edge is ≥ 2, and not assign a particular number to it.

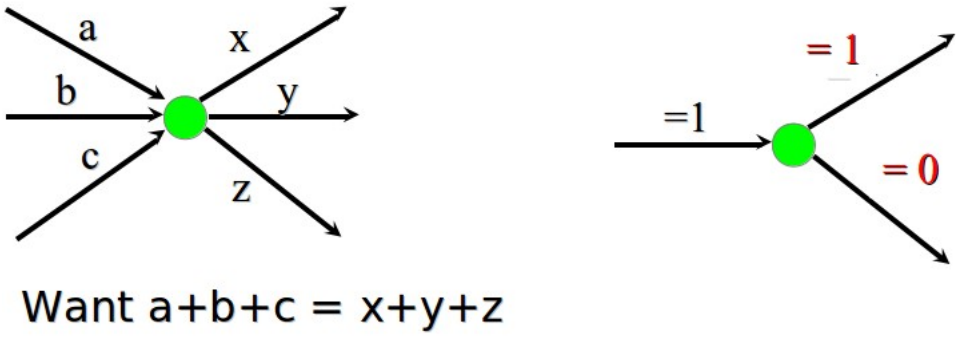


Figure 5.13: *Left:* Flow resolution concept. *Right:* Flow resolution example.

We use reasoning from flows in order to resolve such ambiguities. We need to satisfy the flow constraint at every junction, i.e. the total weight of all the incoming edges must equal the total weight of all the outgoing edges. For example, in the figure 5.13 there is a junction with an incoming edge of weight 1, and two outgoing edges of weight ≥ 0 and ≥ 1. Hence, we can infer that the weights of the outgoing edges are exactly equal to 0 and 1 respectively. A lot of weights can be inferred this way by repetitively applying this same process throughout the entire graph.

5.3.3 Feasible flow

Once we have the graph and the edge weights, we run a min cost flow algorithm on the graph. Since larger genomes may not a have unique min cost flow, we iterative do the following

- Add ϵ penalty to all edges in solution
- Solve flow again - if there is an alternate min cost flow it will now have a smaller cost relative to the previous flow
- Repeat until we find no new edges

After doing the above, we will be able to label each edge as one of the following

- *Required:* edges that were part of all the solutions
- *Unreliable:* edges that were part of some of the solutions
- *Not required:* edges that were not part of any solution

5.3.4 Dealing with sequencing errors

There are various sources of errors in the genome sequencing procedure. Errors are generally of two different kinds, local and global.

Local errors include insertions, deletions and mutations. Such local errors are dealt with when we are looking for overlapping reads. That is, while checking whether reads overlap, we check for overlaps while being tolerant towards sequencing errors. Once we have computed overlaps, we can derive a consensus by mechanisms such as removing indels and mutations that are not supported by any other read and are contradicted by at least 2.

Global errors are caused by other mechasisms such as two different sequences combining together before being read, and hence we get a read which is from different places in the genome. Such reads are called chimers. These errors are resolved while looking for a feasible flow in the network. When the edge corresponding to the chimer is in use, the amount of flow going through this edge is smaller compared to the flow capacity. Hence, the edge can be detected and then ignored.

Each step of the algorithm is made as robust and resilient to sequencing errors as possible. And the number of DNAs split and sequenced is decided in a way so that we are able to construct most of the DNA (i.e. fulfill some quality assurance such as 98% or 95%).

5.3.5 Resources

Some popular genome assemblers using String Graphs are listed below

- Euler (Pevzner, 2001/06) : Indexing → deBruijn graphs → picking paths → consensus
- Valvel (Birney, 2010) : Short reads → small genomes → simplification → error correction
- ALLPATHS (Gnerre, 2011) : Short reads → large genomes → jumping data → uncertainty

5.4 Whole-Genome Alignment

Once we have access to whole-genome sequences for several different species, we can attempt to align them in order to infer the path that evolution took to differentiate these species. In this section we discuss some of the methods for performing whole-genome alignments between multiple species.

5.4.1 Global, local, and 'glocal' alignment

The Needleman-Wunsch algorithm discussed in chapter 2 is the best way to generate an optimal alignment between two or more genome sequences of limited size. At the level of whole genomes, however, the $O(n^2)$ time bound is impractical. Furthermore, in order to find an optimal alignment between k different species, the time for the Needleman-Wunsch algorithm is extended to $O(n^k)$. For genomes that are millions of bases long, this run time is prohibitive (Figure 5.14).

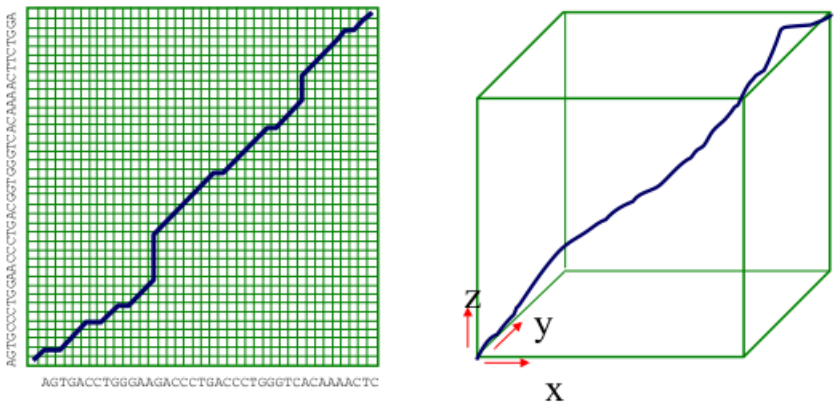


Figure 5.14: The Needleman-Wunsch algorithm for alignments of 2 and 3 genomes

One alternative is to use an efficient local alignment tool such as BLAST to find all of the local alignments, and then chain them together along the diagonal to form global alignments. This approach can save a significant amount of time, since the process of finding local alignments is very efficient, and then we only need to perform the time-consuming Needleman-Wunsch algorithm in the small rectangles between local alignments (Figure 5.15).

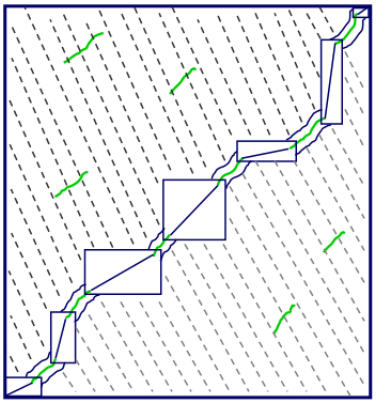


Figure 5.15: We can save time when performing a global alignment by first finding all the local alignments and then chaining them together along the diagonal with restricted dynamic programming

Another novel approach to whole genome alignment is to extend the local alignment search to include inversions, duplications and translocations. Then we can chain these elements together using the least-cost transformations between sequences. This approach is commonly called glocal alignment, since it seeks to combine the best of local and global alignment to create the most accurate picture of how genomes evolve over time (Figure 5.16).

5.4.2 Lagan: Chaining local alignments

LAGAN is a popular software toolkit that incorporates many of the above ideas and can be used for local, global, glocal, and multiple alignments between species.

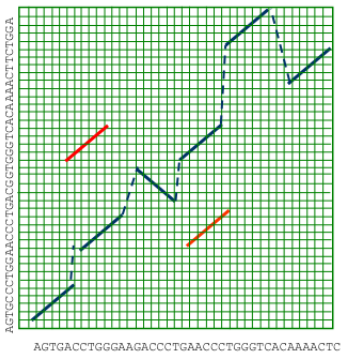


Figure 5.16: Glocal alignment allows for the possibility of duplications, inversion, and translocations

The regular LAGAN algorithm consists of finding local alignments, chaining local alignments along the diagonal, and then performing restricted dynamic programming to find the optimal path between local alignments.

Multi-LAGAN uses the same approach as regular LAGAN but generalizes it to multiple species alignment. In this algorithm, the user must provide a set of genomes and a corresponding phylogenetic tree. Multi-LAGAN performs pairwise alignment guided by the phylogenetic tree. It first compares highly related species, and then iteratively compares more and more distant species.

Shuffle-LAGAN is a glocal alignment tool that finds local alignments, builds a rough homology map, and then globally aligns each of the consistent parts (Figure 5.17). In order to build a homology map, the algorithm chooses the maximum scoring subset of local alignments based on certain gap and transformation penalties, which form a non-decreasing chain in at least one of the two sequences. Unlike regular LAGAN, all possible local alignment sequences are considered as steps in the glocal alignment, since they could represent translocations, inversions and inverted translocations as well as regular untransformed sequences. Once the rough homology map has been built, the algorithm breaks the homologous regions into chunks of local alignments that are roughly along the same continuous path. Finally, the LAGAN algorithm is applied to each chunk to link the local alignments using restricted dynamic programming.

5.4.3 Building Rearrangement graphs

By running S-LAGAN or other glocal alignment tools, we can discover inversions, translocations, and other homologous relations between different species. By mapping the connections between these rearrangements, we can gain insight into how each species evolved from the common ancestor (Figure 5.18).

5.5 Gene-based region alignment

An alternative way for aligning multiple genomes anchors genomic segments based on the genes that they contain, and uses the correspondence of genes to resolve corresponding regions in each pair of species. A nucleotide-level alignment is then constructed based on previously-described methods in each multiply-conserved region.

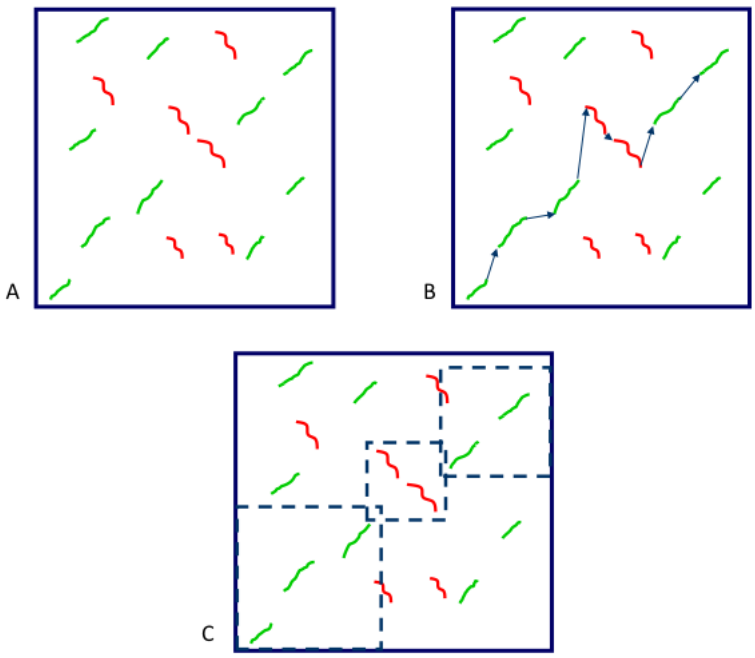


Figure 5.17: The steps to run the SLAGAN algorithm are A. Find all the local alignments, B. Build a rough homology map, and C. globally align the consistent parts using the regular LAGAN algorithm

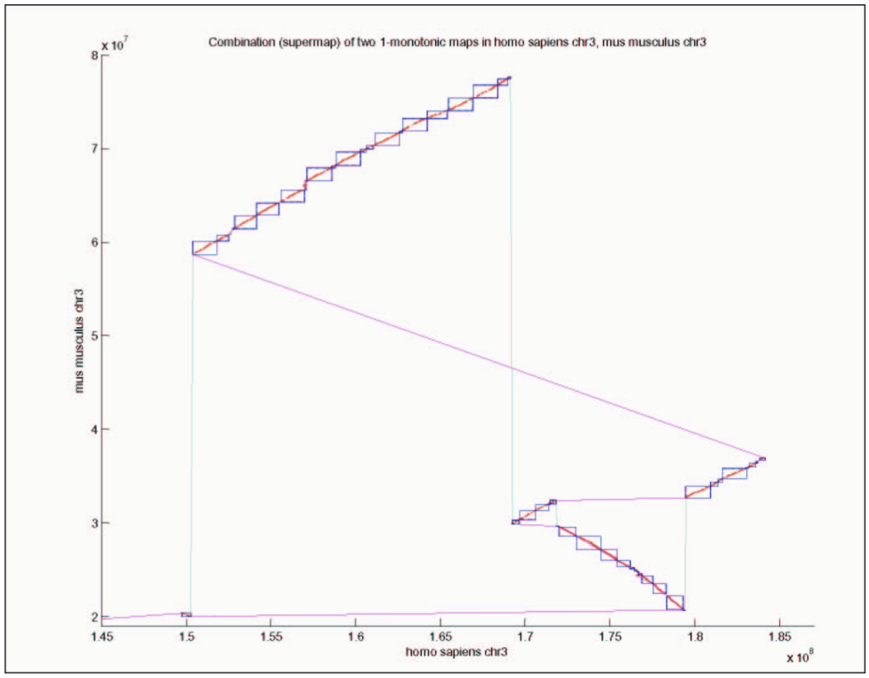


Figure 5.18: Using the concepts of glocal alignment, we can discover inversions, translocations, and other homologous relations between different species such as human and mouse

What makes it difficult is that not all regions have one-to-one correspondence and the sequence is not static genes undergo divergence, duplication, and losses and whole genomes undergo rearrangements. To help overcome these challenges, researchers look at the amino-acid similarity of gene pairs across genomes

and the locations of genes within each genome.

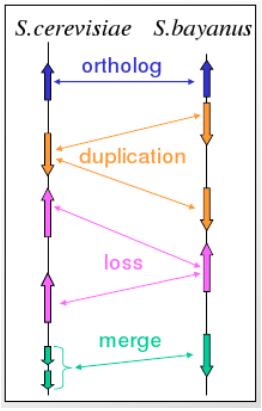


Figure 5.19: Graph of *S. cerevisae* and *S. bayanus* gene correspondence.

Gene correspondence can be represented by a weighted bipartite graph with nodes representing genes with coordinates and edges representing weighted sequence similarity (Figure 5.19). Orthologous relationships are one-to-one matches and paralogous relationships are one-to-many or many-to-many matches. The graph is first simplified by eliminating spurious edges and then edges are selected based on available information such as blocks of conserved gene order and protein sequence similarity.

The Best Unambiguous Subgroups (BUS) algorithm can then be used to resolve the correspondence of genes and regions. BUS extends the concept of best-bidirectional hits and used iterative refinement with a increasing relative threshold. It uses the complete bipartite graph connectivity with integrated amino-acid similarity and gene order information.

In the example of a correctly resolved gene correspondence of *S.cerevisiae* with three other related species, more than 90% of the genes had a one-to-one correspondence and regions and protein families of rapid change were identified.

5.6 Mechanisms of Genome Evolution

Looking at a whole genome, we find that there are specific regions of rapid evolution. In *S. cerevisiae*, for example, 80% of ambiguities are found in 5% of the genome. Telomeres are repetitive DNA sequences at the end of chromosomes which protect the ends of the chromosomes from deterioration. Telomere regions are inherently unstable, tending to undergo rapid structural evolution, and the 80% of variation corresponds to 31 of the 32 telomeric regions. Gene families contained within these regions such as HXT, FLO, COS, PAU, and YRF show significant evolution in number, order, and orientation. Several novel and protein-coding sequences can be found in these regions. Since aside from the telomeric regions, very few genomic rearrangements are found in *S. cerevisiae*, regions of rapid change can be identified by protein family expansions in chromosome ends.

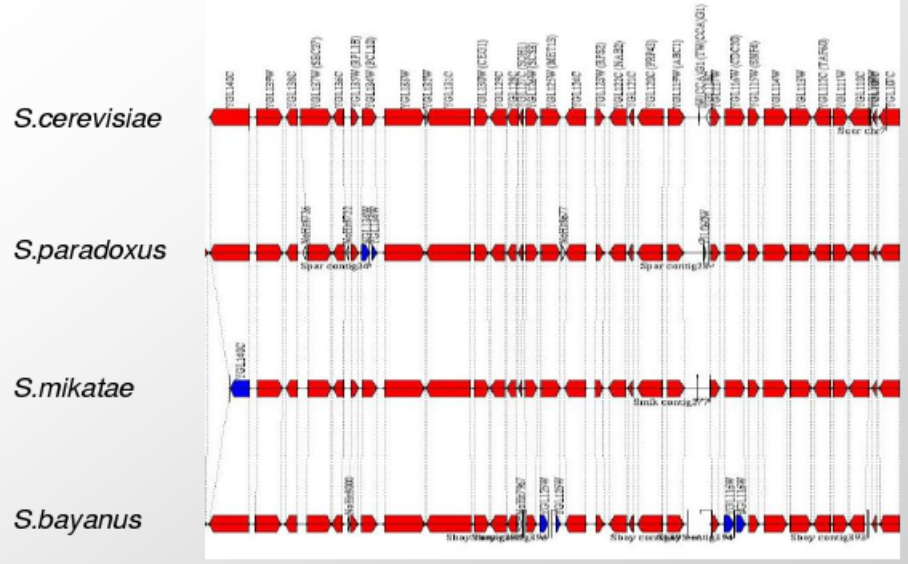


Figure 5.20: Illustration of gene correspondence for *S.cerevisiae* Chromosome VI (250-300bp).

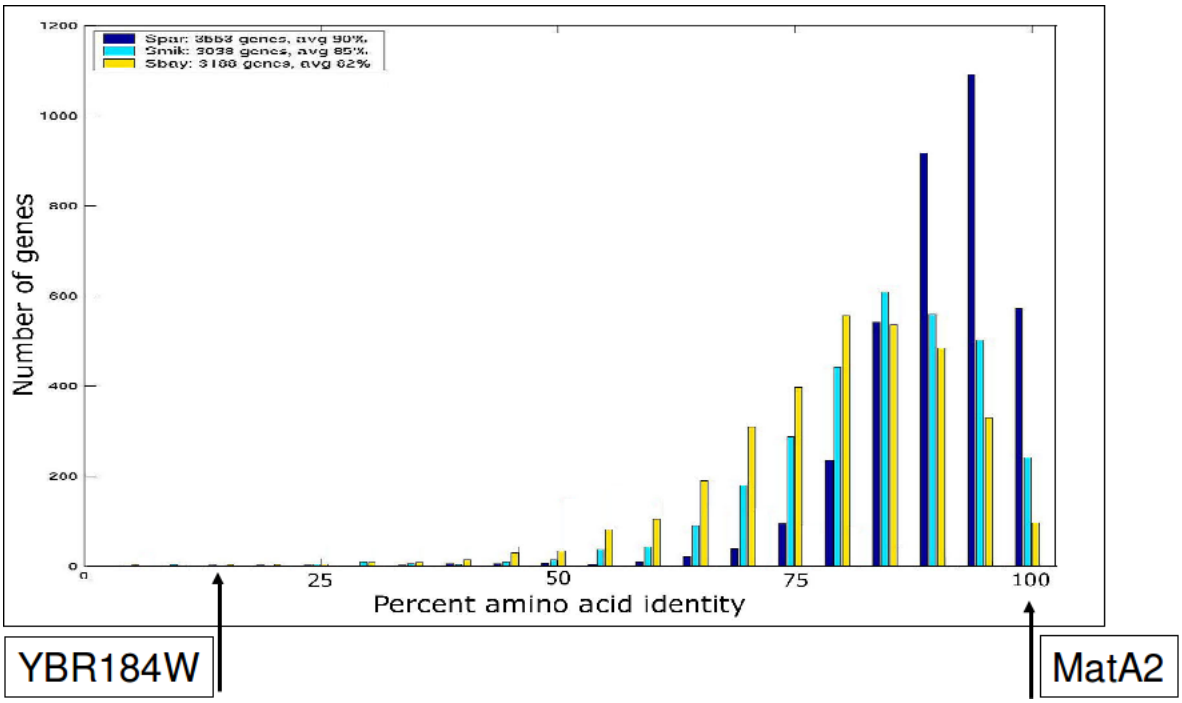


Figure 5.21: Dynamic view of a changing gene

Globally speaking, genes evolve at different rates: there are both fast and slow evolving genes. For example as illustrated in Figure 5.21, on one extreme, there is YBR184W in yeast which shows unusually low sequence conservation and exhibits numerous insertions and deletions across species. On the other extreme there is MatA2, which shows perfect amino acid and nucleotide conservation. Mutation rates often also vary by functional classification. For example, mitochondrial ribosomal proteins are less conserved than ribosomal proteins.

The fact that some genes evolve more slowly in one species versus another may be due to factors such as longer life cycles. Lack of evolutionary change in specific genes, however, suggests that there are additional biological functions which are responsible for the pressure to conserve the nucleotide sequence. Yeast can switch mating types by switching all their A and α genes and MatA2 is one of the four yeast mating-type genes (MatA2, Matα2, MatA1, Matα1). Its role could potentially be revealed by nucleotide conservation analysis.

Fast evolving genes can also be biologically meaningful. Mechanisms of rapid protein change include:

- Protein domain creation via stretches of Glutamine (Q) and Asparagine (N) and protein-protein interactions,
- Compensatory frame-shifts which enable the exploration of new reading frames and reading/creation of RNA editing signals,
- Stop codon variations and regulated read-through where gains enable rapid changes and losses may result in new diversity
- Inteins, which are segments of proteins that can remove themselves from a protein and then rejoin the remaining protein, gain from horizontal transfers of post-translationally self-splicing inteins.

We now look at differences in gene content across different species (*S.cerevisiae*, *S.paradoxus*, *S.mikatae*, and *S.bayanus*.) A lot can be revealed about gene loss and conversion by observing the positions of paralogs across related species and observing the rates of change of the paralogs. There are 8-10 genes unique to each genome which are involved mostly with metabolism, regulation and silencing, and stress response. In addition, there are changes in gene dosage with both tandem and segment duplications. Protein family expansions are also present with 211 genes with ambiguous correspondence. All in all however, there are few novel genes in the different species.

5.6.1 Chromosomal Rearrangements

These are often mediated by specific mechanisms as illustrated for Saccharomyces in Figure5.22.

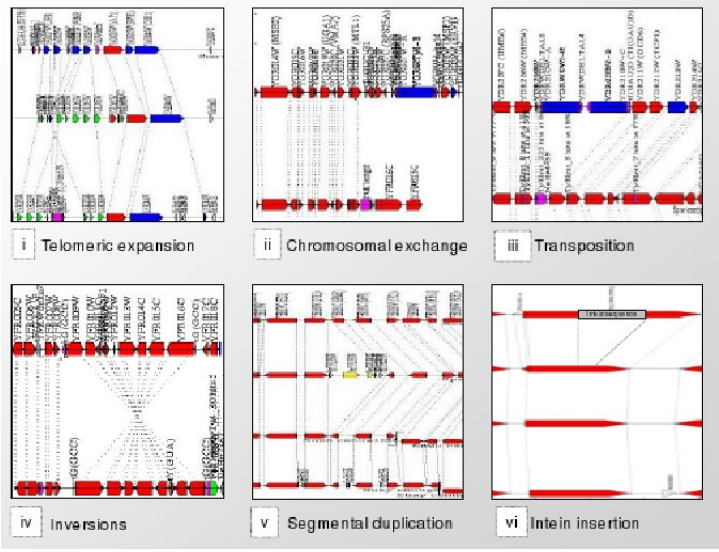


Figure 5.22: Mechanisms of chromosomal evolution.

Translocations across dissimilar genes often occur across transposable genetic elements (Ty elements in yeast for example). Transposon locations are conserved with recent insertions appearing in old locations and long terminal repeat remnants found in other genomes. They are evolutionarily active however (for example with Ty elements in yeast being recent), and typically appear in only one genome. The evolutionary advantage of such locationally conserved transposons may lie in the possibility of mediating reversible arrangements. Inversions are often flanked by tRNA genes in opposite transcriptional orientation. This may suggest that they originate from recombination between tRNA genes.

5.7 Whole Genome Duplication

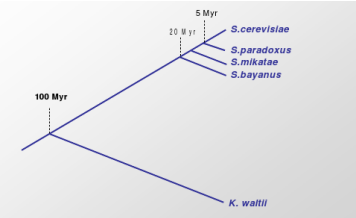


Figure 5.23: Moving further back in evolutionary time for Saccharomyces.

As you trace species further back in evolutionary time, you have the ability to ask different sets of questions. In class, the example used was *K. waltii*, which dates to about 95 millions years earlier than *S.cerevisiae* and 80 million years earlier than *S.bayanus*.

Looking at the dotplot of *S.cerevisiae* chromosomes and *K.waltii* scaffolds, a divergence was noted along the diagonal in the middle of the plot, whereas most pairs of conserved region exhibit a dot plot with a clear and straight diagonal. Viewing the segment at a higher magnification (Figure 5.24), it seems that *S.cerevisiae* sister fragments all map to corresponding *K.waltii* scaffolds.

Schematically (Figure 5.25) sister regions show gene interleaving. In duplicate mapping of centromeres, sister regions can be recognized based on gene order. This observed gene interleaving provides evidence of complete genome duplication.

5.8 Additional figures

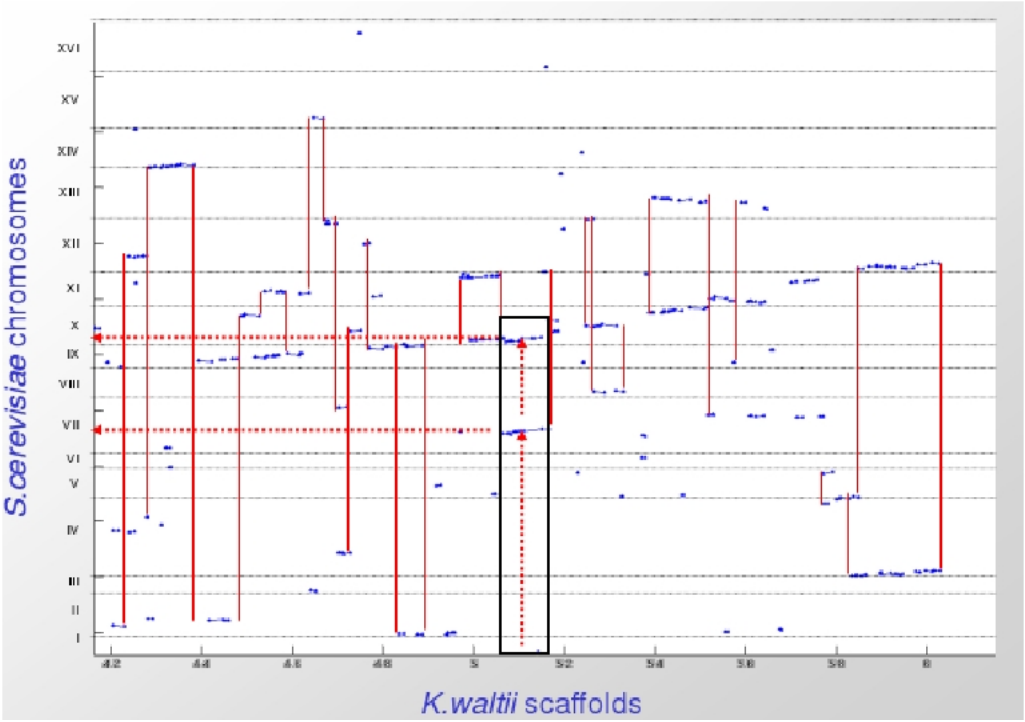


Figure 5.24: Gene Correspondence for *S.cerevisiae* chromosomes and *K.waltii* scaffolds.

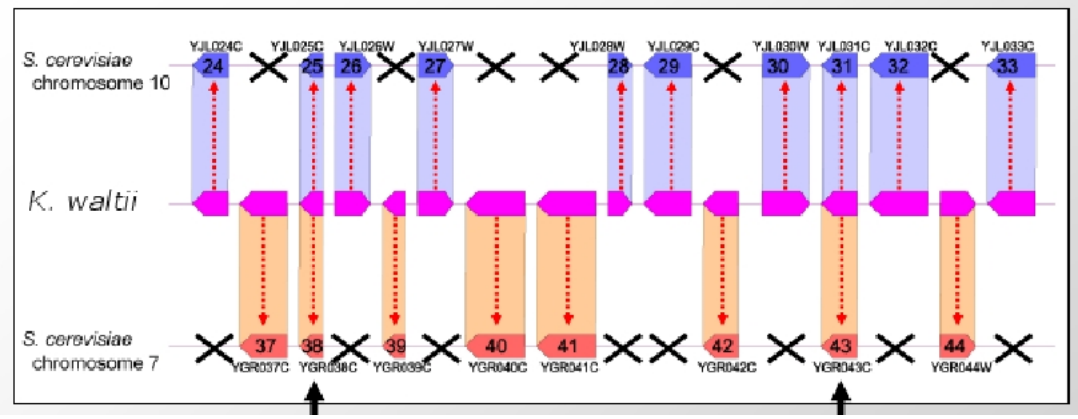


Figure 5.25: Gene interleaving shown by sister regions in *K.waltii* and *S.cerevisae*

Bibliography

[1] Embl allextron database - cassette exons.

[2] Batzoglou S et al. Arachne: a whole-genome shotgun assembler. *Genome Res*, 2002.

[3] Manolis Kellis. Lecture slides 04: Comparative genomics i. September 21,2010.

[4] Manolis Kellis. Lecture slides 05.1: Comparative genomics ii. September 23, 2010.

[5] Manolis Kellis. Lecture slides 05.2: Comparative genomics iii, evolution. September 25,2010.

[6] Nikolaus Rajewsky Kevin Chen. The evolution of gene regulation by transcription factors and microrna. *Nature Reviews Genetics*, 2007.

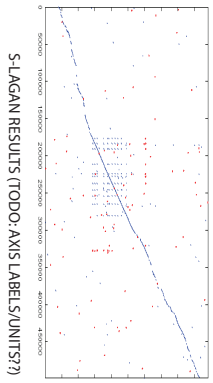


Figure 5.26: S-LAGAN results.

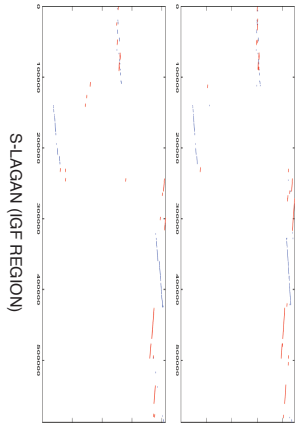


Figure 5.27: S-LAGAN results for IGF locus.

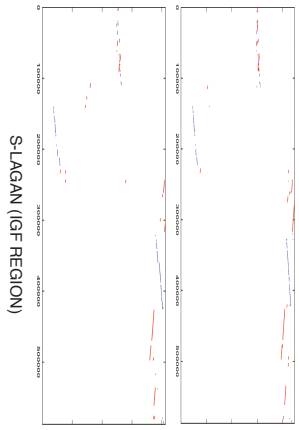


Figure 5.28: S-LAGAN results for IGF locus.

[7] Douglas Robinson and Lynn Cooley. Examination of the function of two kelch proteins generated by stop codon suppression. *Development*, 1997.

[8] Stark. Discovery of functional elements in 12 drosophila genomes using evolutionary signatures. *Nature*, 2007.

[9] Angela Tan. Lecture 15 notes: Comparative genomics i: Genome annotation. November 4, 2009.

CHAPTER

SIX

BACTERIAL GENOMICS– MOLECULAR EVOLUTION AT THE LEVEL OF ECOSYSTEMS

Guest Lecture by Eric Alm
Scribed by Deniz Yorukoglu (2011)

Figures

6.1 Introduction

With the magnitude and diversity of bacterial populations in human body, human microbiome has many common properties with natural ecosystems researched in environmental biology. As a field with a large number of quantitative problems to tackle, bacterial genomics offers an opportunity for computational biologist to be actively involved in the progress of this research area.

There are approximately 10^{14} microbial cells in an average human gut, whereas there are only 10^{13} human cells in a human body in total. Furthermore, there are 10^{12} external microbial cells living on our skin. From a cell count perspective, this corresponds to 10 times more bacterial cells in our body than our own cells. From a gene count perspective, there are 100 times more genes belonging to the bacteria living in/on us than to our own cells. For this reason, these microbial communities living in our bodies are an integral part of

what makes us human and we should research upon these genes that are not directly encoded in our genome, but still have a significant effect on our physiology.

6.1.1 Evolution of microbiome research

Earlier stages of microbiome research were mostly based on data collection and analysis of surveys of bacterial groups present in a particular ecosystem. Apart from collecting data, this type of research also involved sequencing of bacterial genomes and identification of gene markers for determining different bacterial groups present in the sample. The most commonly used marker for this purpose is 16S rRNA gene, which is a section of the prokaryotic DNA that codes for ribosomal RNA. Three main features of 16S gene that makes it a very effective marker for microbiome studies are: (1) its short size (∼1500 bases) that makes it cheaper to sequence and analyze, (2) high conservation due to exact folding requirements of the ribosomal RNA it encodes for, and (3) its specificity to prokaryote organisms that allows us to differentiate from contaminant protist, fungal, plant and animal DNAs.

A further direction in early microbial research was inferring rules from generated datasets upon microbial ecosystems. These studies investigated initially generated microbial data and tried to understand rules of microbial abundance in different types of ecosystems and infer networks of bacterial populations regarding their co-occurrence, correlation and causality with respect to one another.

A more recent type of microbial research takes a predictive approach and aims to model the change of bacterial populations in an ecosystem through time making use of differential equations. For example, we can model the rate of change for the population size of a particular bacterial group in human gut as an ordinary differential equation (ODE) and use this model to predict the size of the population at a future time point by integrating over the time interval.

We can further model change of bacterial populations with respect to multiple parameters, such as time and space. When we have enough data to represent microbial populations temporally and spatially, we can model them using partial differential equations (PDEs) for making predictions using multivariate functions.

6.1.2 Data generation for microbiome research

Data generation for microbiome research usually follows the following work-flow: (1) a sample of microbial ecosystem is taken from the particular site being studied (e.g. a patient's skin or a lake), (2) the DNAs of the bacteria living in the sample are extracted, (3) 16S rDNA genes are sequenced, (4) conserved motifs in some fraction of the 16S gene (DNA barcodes) are clustered into **operational taxonomic units** (OTUs), and (5) a vector of abundance is constructed for all species in the sample. In microbiology, bacteria are classified into OTUs according to their functional properties rather than species, due to the difficulty in applying the conventional species definition to the bacterial world.

In the remainder of the lecture, a series of recent studies that are related to the field of bacterial genomics and human microbiome studies are described.

6.2 Study 1: Evolution of life on earth

This study [2] is inspired from a quote by Max Delbruck: "Any living cell carries with it the experience of a billion years of experimentation by its ancestors". In this direction, it is possible to find evidence in the genomes of living organisms for ancient environmental changes with large biological impacts. For instance, the oxygen that most organisms currently use would have been extremely toxic to almost all life on earth before the accumulation of oxygen via oxygenic photosynthesis. It is known that this event happened approximately 2.4 billion years ago and it caused a dramatic transformation of life on earth.

A dynamic programming algorithm was developed in order to infer gene birth, duplication, loss and horizontal gene transfer events given the phylogeny of species and phylogeny of different genes. **Horizontal gene transfer** is the event in which bacteria transfer a portion of their genome to other bacteria from different taxonomic groups.

Figure 6.1 shows an overview of these inferred events in a phylogenetic tree focusing on prokaryote life. In each node, the size of the pie chart represents the amount of genetic change between two branches and each colored slice stands for the rate of a particular genetic modification event. Starting from the root of the

tree, we see that almost the entire pie chart is represented by newly born genes represented by red. However, around 2.5 billion years ago green and blue slices become more prevalent, which represent rate of horizontal gene transfer and gene duplication events.

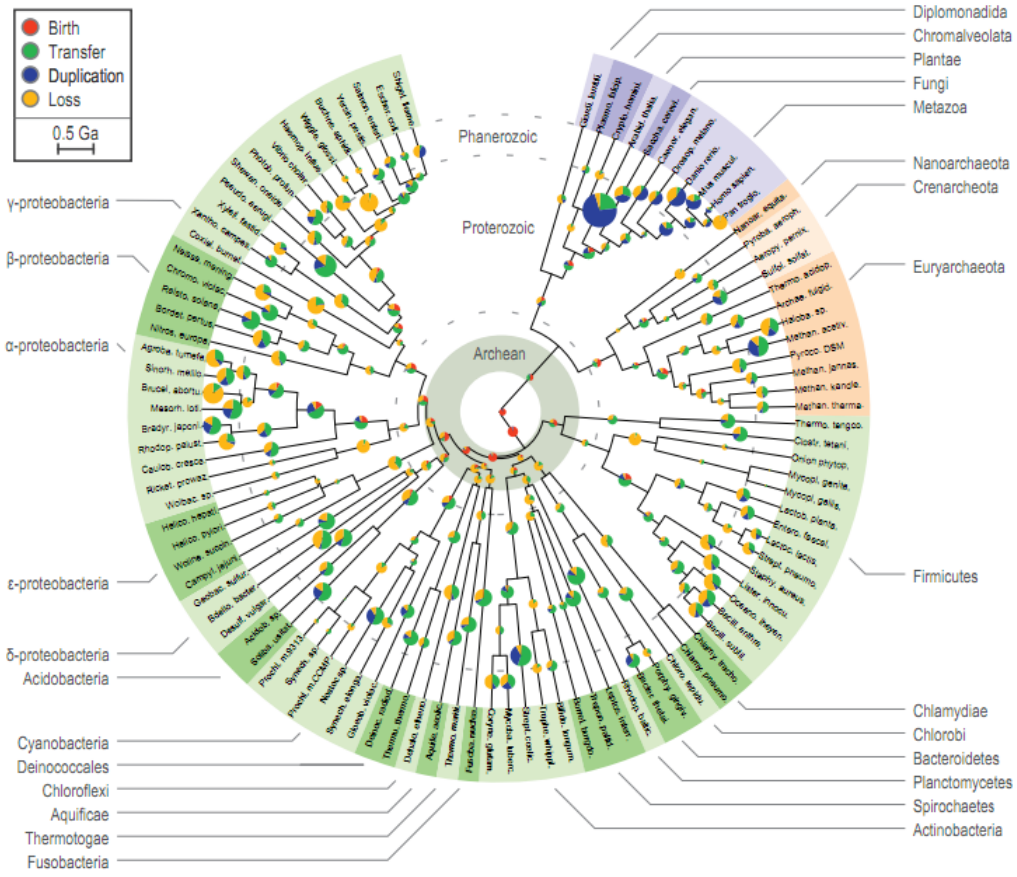


Figure 6.1: A tree of life displaying rates of gene birth, duplication, loss, and horizontal gene transfer at each branching point.

In Figure 6.2, a large spike can be seen during Archean eon representing large amount of genetic change on earth occurring during this particular time period. This study looked for enzymatic activity of genes that were born in this eon different from the genes that were already present. On the right hand side of Figure 6.2, logarithmic enrichment levels of different metabolites are displayed. Most enriched metabolites produced by these genes were discovered to be functional in oxidation reduction and electron transport. Overall, this study suggests that life invented modern electron transport chain around 3.3 billion years ago and around 2.8 billion years ago organisms evolved to use the same proteins that are used for producing oxygen also to breathe oxygen.

6.3 Study 2: Pediatric IBD study with Athos Boudvaros

In some diseases such as Inflammatory Bowel Disease (IBD); if the disease is not diagnosed and monitored closely, the results can be very severe, such as the removal of the patient's colon. On the other hand, currently existing most reliable diagnosis methods are very invasive (e.g. colonoscopy). An alternative approach for diagnosis can be abundance analysis of the microbial sample taken from the patients' colon. This study aims to predict the disease state of the subject from bacterial abundances in stool samples taken from the patient.

105 samples were collected for this study among the patients of Dr. Athos Boudvaros; some of them

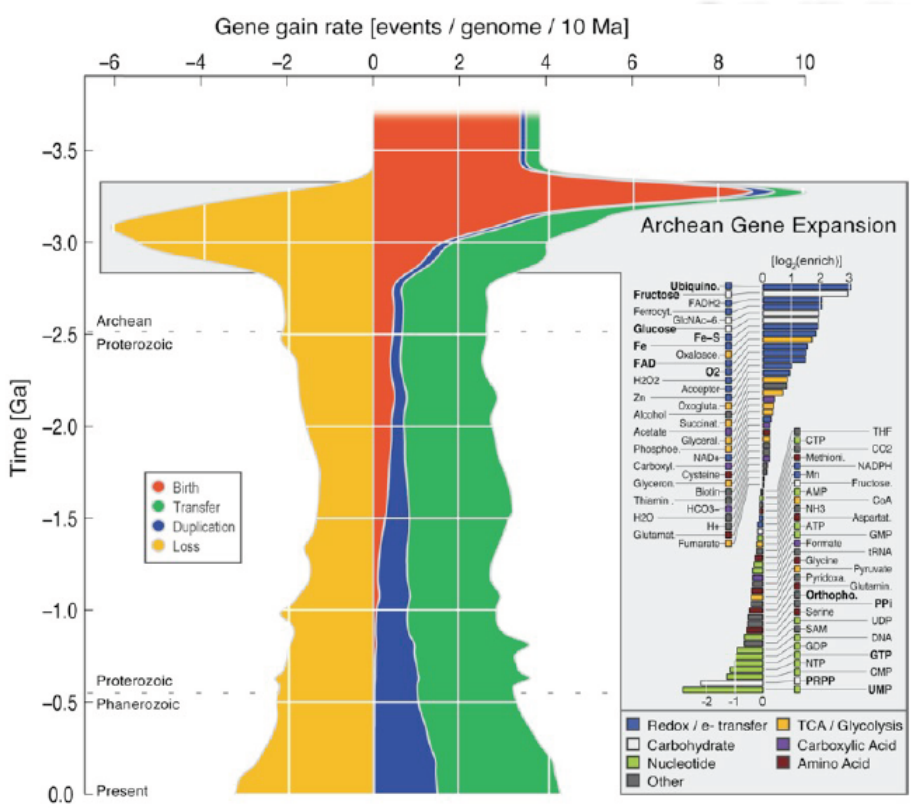


Figure 6.2: Rates of new gene birth, duplication, loss and horizontal gene transfer during Archean Gene Expansion

displaying IBD symptoms and others different diseases (control group). In Figure 6.3, each row block represents a set of bacterial groups at a taxonomic level (phylum level at the top and genus level at the bottom) and each column block represents a different patient group: control patients, Crohn's disease (CD), and ulcerative colitis (UC). The only significant single biomarker was E. Coli, which is not seen in control and CD patients but seen in about a third of the UC patients. There seems to be no other single bacterial group that gives significant classification between the patient groups from these abundance measures.

Since E. Coli abundance is not a clear-cut single bacterial biomarker, using it as a diagnostic tool would yield low accuracy classification. On the other hand, we can take the entire bacterial group abundance distribution and feed them into a random forest and estimate cross-validation accuracy. After the classification method was employed, it was able to tell with 90% accuracy if the patient is diseased or not. This suggests that it is a competitive method with respect to other non-invasive diagnotic approaches which are generally highly specific but not sensitive enough.

One key difference between control and disease groups is the decrease in the diversity of the ecosystem. This suggests that the disease status is not controlled by a single germ but the overall robustness and the resilience of the ecosystem. When diversity in the ecosystem decreases, the patient might start showing disease symptoms.

6.4 Study 3: Human Gut Ecology (HuGE) project

This study aims to identify more than three hundred dietary and environmental factors affecting human microbiome. The factors, which were regularly tracked by an iPhone App, were the food the subject ate, how much they slept, the mood they were in etc. Moreover, stool samples were taken from the subjects every day for a year in order to perform sequence analysis of the bacterial group abundances for a specific day

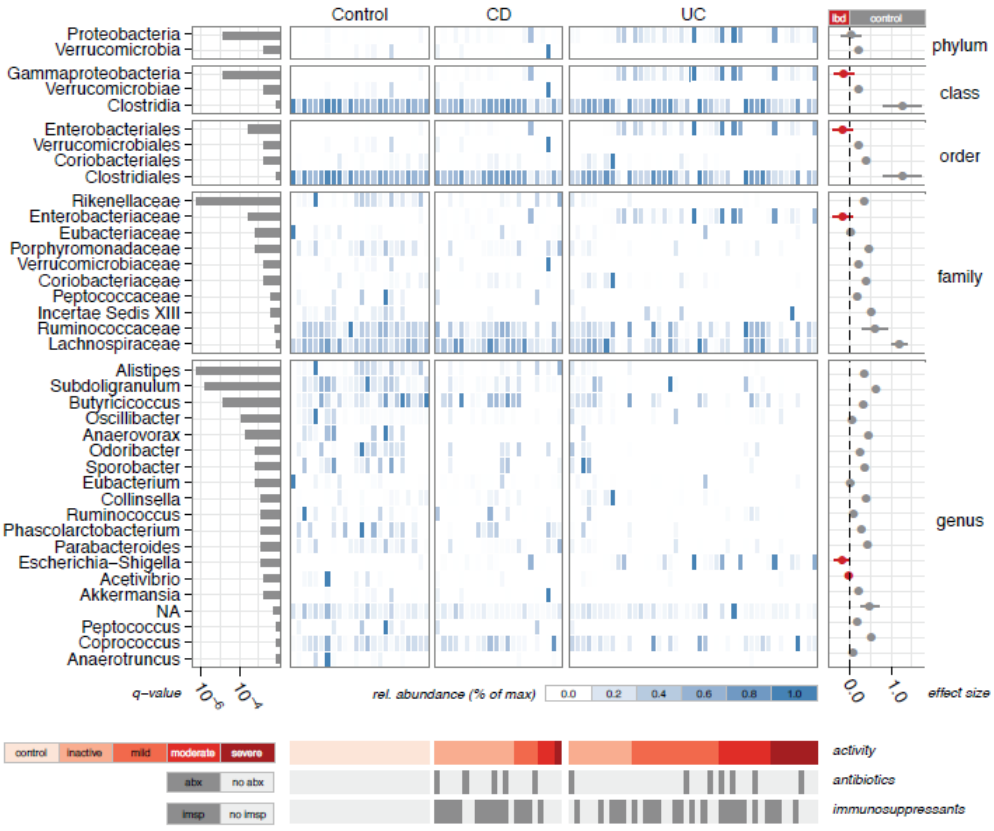


Figure 6.3: Abundance levels of different bacterial groups in control patients, Crohn's disease patients and patients with ulcerative colitis.

relevant to a particular environmental factor. The motivation behind carrying out this study is that, it is usually very hard to get a strong signal between bacterial abundances and disease status. Exploring dietary effects on human microbiome might potentially elucidate some of these confounding factors in bacterial abundance analysis. However, this study analyzed dietary and environmental factors on only two subjects' gut ecosystems; inferring statistically significant correlations with environmental factors would require large cohorts of subjects.

Figure 6.4 shows abundance levels of different bacterial groups in the gut of the two donors throughout the experiment. One key point to notice is that within an individual, the bacterial abundance is very similar through time. However, bacterial group abundances in the gut significantly differ from person to person.

One statistically significant dietary factor that was discovered as a predictive marker for bacterial population abundances is fiber consumption. It was inferred that fiber consumption is highly correlated with the abundance of bacterial groups such as Lachnospiraceae, Bifidobacteria, and Ruminococcaceae. In Donor B, 10g increase in fiber consumption increased the overall abundance of these bacterial groups by 11%.

In Figure 6.6 and Figure 6.7, a horizon plot of the two donors B and A are displayed respectively. A legend to read these horizon plots is given in Figure 6.5. For each bacterial group the abundance-time graph is displayed with different colors for different abundance layers, segments of different layers are collapsed into the height of a single layer displaying only the color with the highest absolute value difference from the normal abundance, and finally the negative peaks are switched to positive peaks preserving their original color.

In Figure 6.6, we see that during the donor's trip to Thailand, there is a significant change in his gut bacterial ecosystem. A large number of bacterial groups disappear (shown on the lower half of the horizon plot) as soon as the donor starts living in Thailand. And as soon as the donor returns to U.S., the abundance

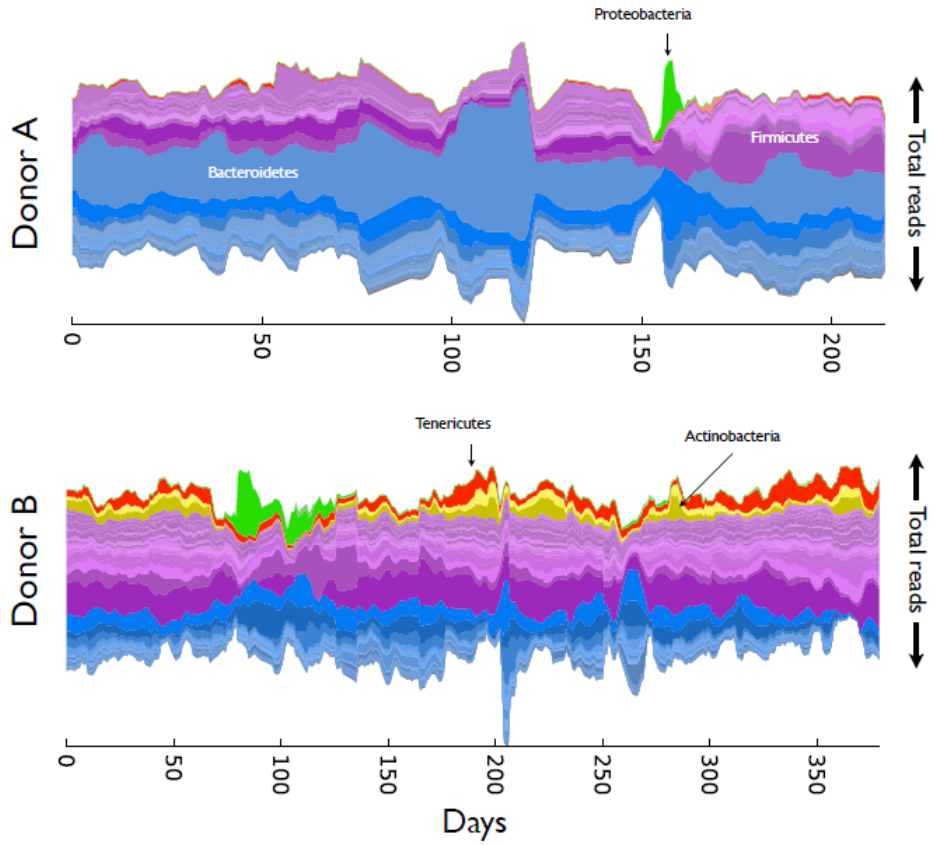


Figure 6.4: Gut bacterial abundances plotted through time for the two donors participating in HuGE project.

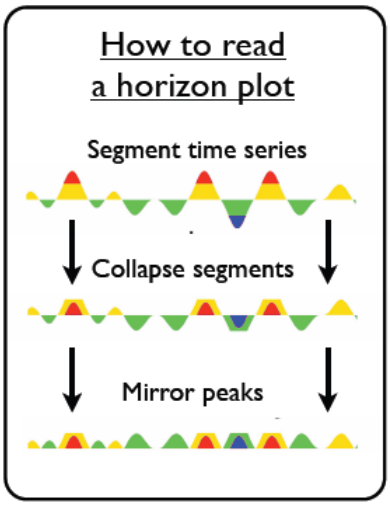


Figure 6.5: Description of how to read a horizon plot.

levels of these bacterial groups quickly return back to their normal levels. Moreover, some bacterial groups that are normally considered to be pathogens (first 8 groups shown on top) appears in the donor's ecosystem almost as soon as the donor moves to Thailand and mostly disappears when he returns back to United States. This indicates that environmental factors (such as location) can cause major changes in our gut ecosystem

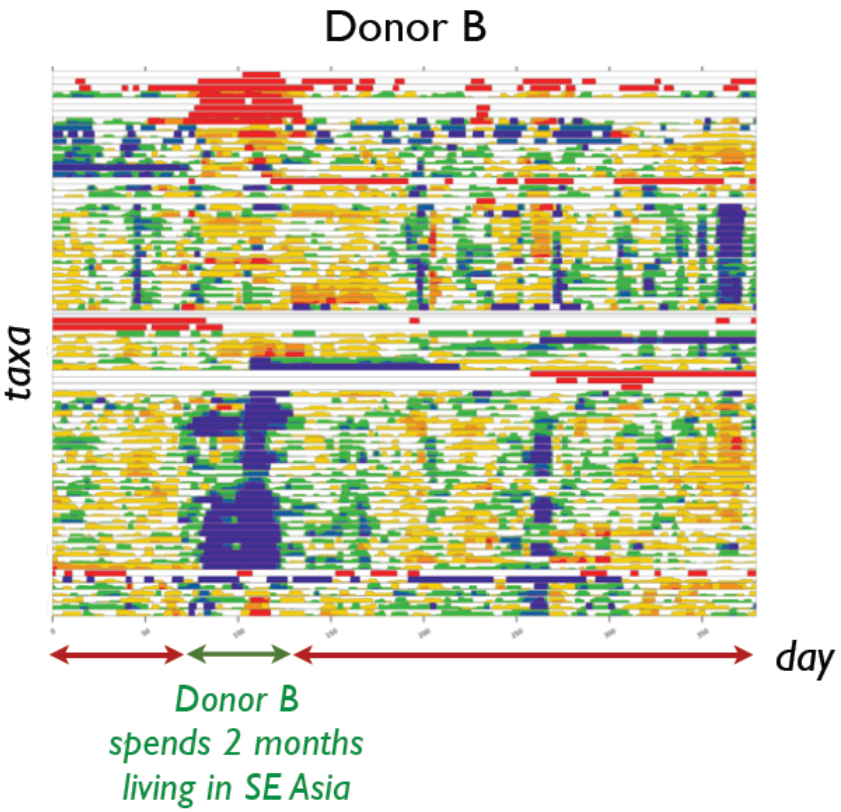


Figure 6.6: Horizon plot of Donor B in HuGE study.

while the environmental factor is present but can disappear after the factor is removed.

In Figure 6.7, we see that after the donor is infected with salmonella, a significant portion of his gut ecosystem is replaced by other bacterial groups. A large number of bacterial groups permanently disappear during the infection and other bacterial groups replace their ecological niches. In other words, the introduction of a new environmental factor takes the bacterial ecosystem in the donor's gut from one equilibrium point to a completely different one. Even though the bacterial population mostly consists of salmonella during the infection, before and after the infection the bacterial count stays more or less the same. The scenario that happened here is that salmonella drove some bacterial groups to extinction in the gut and similar bacterial groups took over their empty ecological niches.

In Figure 6.8, p-values are displayed for day-to-day bacterial abundance correlation levels for Donor A and B. In Donor A's correlation matrix, there is high correlation within the time interval a corresponding to pre-infection and within the time interval b corresponding to post-infection. However, between a and b there is almost no correlation at all. On the other hand, in the correlation matrix of donor B, we see that pre-Thailand and post-Thailand time intervals, c, have high correlation within and between themselves. However, the interval d that correspond to the time period of Donor B's trip to Thailand, we see relatively little correlation to c. This suggests that the perturbations in the bacterial ecosystem of Donor B wasn't enough to cause a permanent shift of the abundance equilibrium as in the case with Donor A due to salmonella infection.

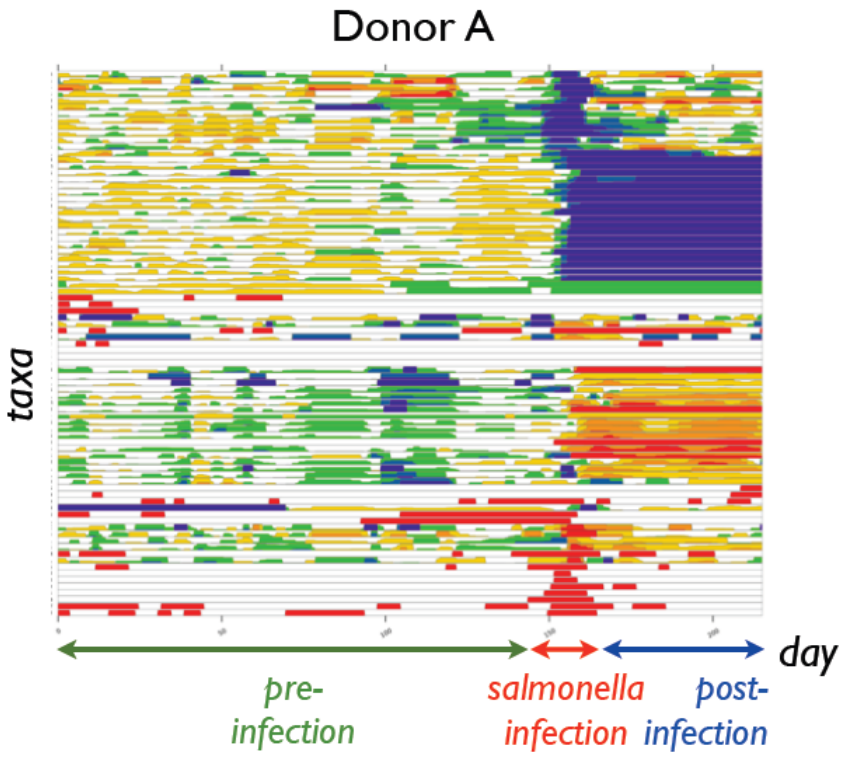


Figure 6.7: Horizon plot of Donor A in HuGE study.

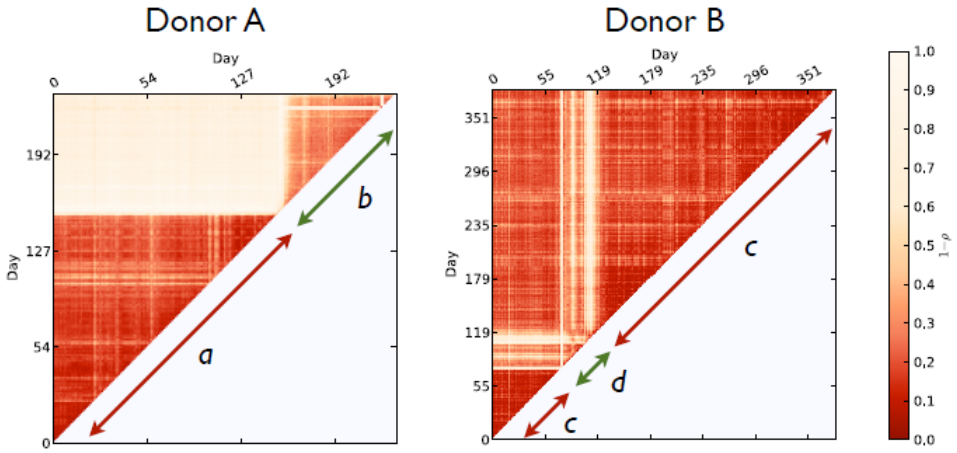


Figure 6.8: Day-to-day bacterial abundance correlation matrices of Donor A and Donor B.

6.5 Study 4: Microbiome as the connection between diet and phenotype

In a study by Mozaffarian et al. [4] more than a hundred thousand patients were analyzed with the goal of discovering the effect of diet and lifestyle choices on long-term weight gain and obesity. This study built a model to predict the patients' weights based on the types and amounts of food they consumed over a certain period of time. They found out that fast-food type of food (processed meats, potato chips, sugar-sweetened

beverages) were were most highly correlated with obesity. On the other hand, consumption level of yogurt was inversely correlated with obesity.

Further experiments with mouse and human cohorts showed that, within both control group and fast-food group, increased consumption of yogurt leads to weight loss. In the experiment with mice, some female mice were given Lactobacillus reuteri (a group of bacteria found in yogurt) and allowed to eat as much regular food or fast-food they wanted to. This resulted in significant weight loss in the group of mice that were given the purified bacterial extract.

An unexpected phenotypical effect of organic yogurt consumption was discovered to be shinier coat of the mice and dogs that were given yogurt as part of their diet. A histological analysis of the skin biopsy of the control and yogurt fed mice proves that the mice that were fed the bacteria in yogurt had hair follicles that are active, leading to active development of healthier and shiny coat and hair.

6.6 Study 5: Horizontal Gene Transfer (HGT) between bacterial groups and its effect on antibiotic resistance

A study by Hehemann et al. [3] discovered a specific gene that digests a type of sulfonated carbohydrate that is only found in seaweed sushi wrappers. This gene is found in the gut microbes of Japanese people but not North Americans. The study concluded that this specific gene has transferred at some point in history from the algae itself to the bacteria living on it and then to the gut microbiome of a Japanese person by horizontal gene transfer. This study also suggests that, even though some bacterial group might live in our gut for our entire lives, they can gain new functionalities throughout our lives by picking up new genes depending on the type of food that we eat.

In this direction, a study in Alm's Laboratory investigated around 2000 bacterial genomes published in [1] with the aim of detecting genes that are 100% similar but belong to bacteria in different taxonomic groups. Any gene that is exactly the same between different bacterial groups would indicate a horizontal gene transfer event. In this study, around 100000 such instances were discovered.

When looked at specific environments, it was discovered that the bacteria isolated from humans share genes mostly with other bacteria isolated from human sites. If we focus on more specific sites; we see that bacterial genomes isolated from human gut share genes mostly with with other bacteria that are isolated from gut, and bacterial genomes isolated from human skin shared gene mostly with other isolated from human skin. This finding suggests that independent from the phylogeny of the bacterial groups, ecology is the most important factor determining the amount of gene transfer instances between bacterial groups.

In Figure 6.9, we see that between different bacterial groups taken from human that has at least 3% 16S gene distance, there is around 23% chance that they will share an identical gene in their genome. Furthermore, there is more than 40% chance that they share an identical gene if they are sampled from the same site as well.

On the other hand, Figure 6.10 shows that geography is a weak influence on horizontal gene transfer. Bacterial populations sampled from the same continent and different continents had little difference in terms of the amount of horizontal gene transfer detected.

Figure 6.11 shows a color coded matrix of the HGT levels between various human and non-human environments; top-right triangle representing the amount of horizontal gene transfers and the bottom-left triangle showing the percentage of antibiotic resistance (AR) genes among the transferred genes. In the top-right corner, we see that there is a slight excess of HGT instances between human microbiome and bacterial samples taken from farm animals. And when we look at the corresponding percentages of antibiotic resistance genes, we see that more than 60% of the transfers are AR genes. This result shows the direct effect of feeding subtherapeutic antibiotics to livestock on the emergence of antibiotic resistance genes in the bacterial populations living in human gut.

6.7 Study 6: Identifying virulence factors in Meningitis

Bacterial meningitis is a disease that is caused by very diverse bacteria that are able to get into the blood stream and cross the blood-brain barrier. This study aimed to investigate the virulence factors that can turn

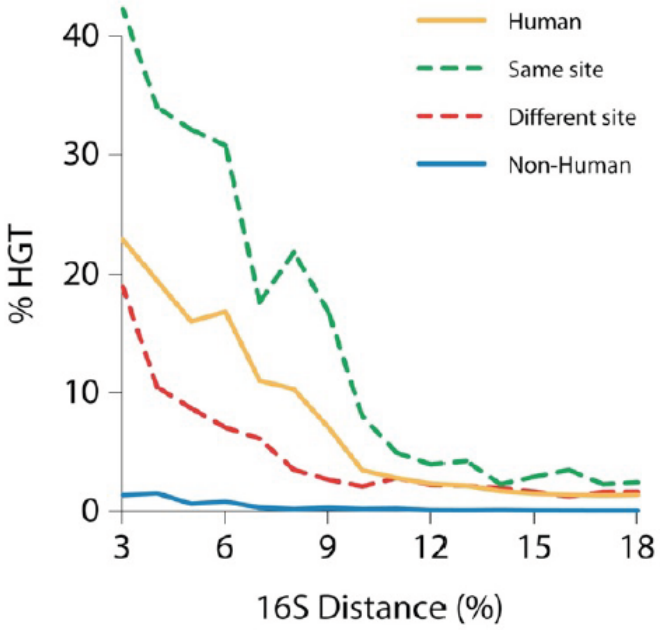


Figure 6.9: Rate of horizontal gene transfer between different bacterial groups taken from non-human sites, human sites, same site within human, and different sites within human.

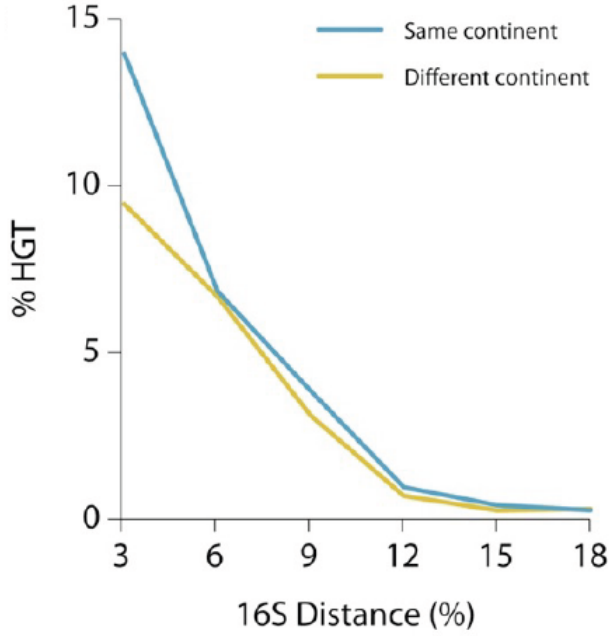


Figure 6.10: Rate of horizontal gene transfer between bacterial groups sampled from the same continent and from different continents.

bacteria into a type that can cause meningitis.

The study involved 70 bacterial strains isolated from meningitis patients, comprising 175172 genes in total. About 24000 of these genes had no known function. There could be some genes among these 24000

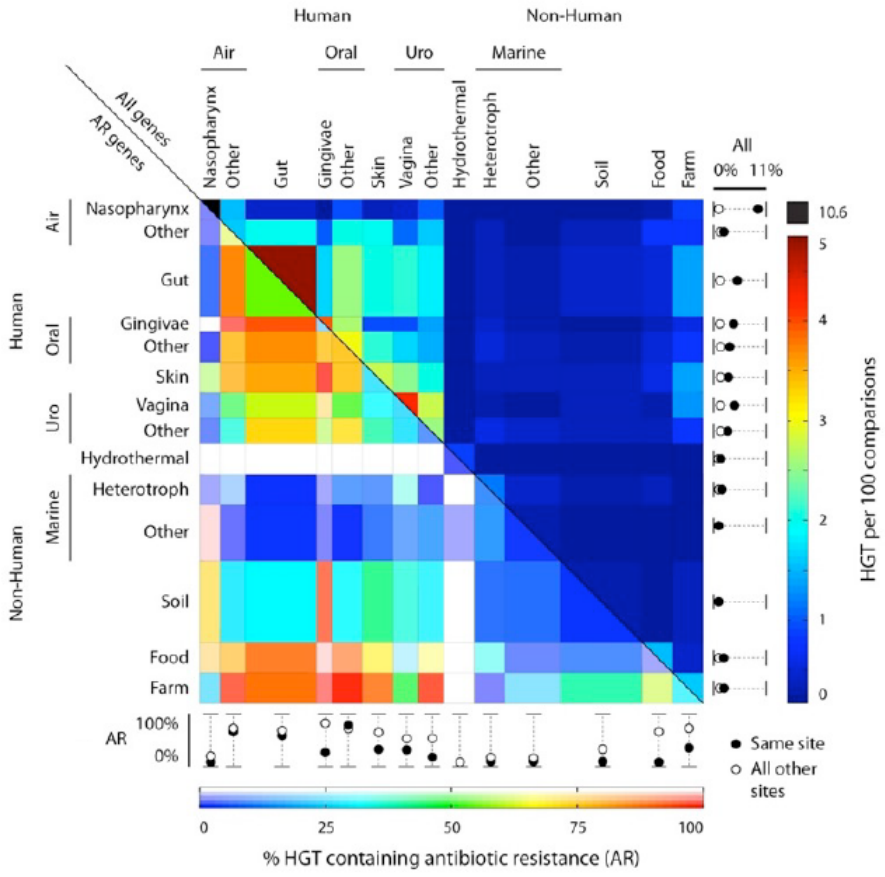


Figure 6.11: Rate of horizontal gene transfer between different human and non-human sites (top right) and the percentage of antiboitic resistance genes among horizonta gene transfers (bottom left).

that might be leading to meningitis causing bacteria and might be good drug targets. Moreover, 82 genes were discovered to be involved in horizontal gene transfer. 69 of these had known functions and 13 of them belonged to the 24000 genes that we do not have any functional information. Among the genes with known function, some of them were related to AR, detoxification, and also some were related to known virulence factors such as hemalysin that lets the bacteria live in the blood stream and adhesin that helps the bacteria latch onto the vein and potentially cross blood brain barrier.

6.8 Q/A

Q: Do you think after some time Donor A in Study 3 will have its bacterial ecosystem return back to its original pre-infection state?

A: The salmonella infection caused certain niches to be wiped out from the bacterial ecosystem of Donor A which were then filled in by similar type of bacteria and reached to a different ecosystem at a new equilibrium. Since these niches are dominated by the new groups of bacteria, it would not be possible for the previous bacterial groups to replace them without a large-scale change in his gut ecosystem.

Q: Is the death of certain bacterial groups in the gut during salmonella infection caused directly by the infection or is it an immune response to cure the disease?

A: It can be both, but it is very hard to tell from the data in Study 3 since it is only a data point that corresponds to the event that we can observe. A future study that tries to figure out what is happening in our immune system during the infection can be observed by drawing blood from the patients during the infection.

Q: Is there a particular connection between an individual's genome and the dominant bacterial groups in the bacterial ecosystem? Would twins show more similar bacterial ecosystems?

A: Twins in general have similar bacterial ecosystems independent from whether they live together or are separated. Even though this seems to be a genetic factor at first, monozygotic and dizygotic twins have the exact same effect, as well as displaying similarity to their mothers' bacterial ecosystem. The reason for this is that starting from birth there is a period of time in which the bacterial ecosystem is programmed. The similarity effect between twins is based on this more than genetic factors.

6.9 Current research directions

A further extension to HuGE study could observe mice gut microbiome during a salmonella infection and observe the process of some bacterial groups being driven to extinction and other types of bacteria replacing the ecological niches that are emptied by them. A higher resolution observation of this phenomenon in mice could illuminate how bacterial ecosystems shift from one equilibrium to another.

6.10 Further Reading

- Overview of Human Microbiome Project: http://commonfund.nih.gov/hmp/overview.aspx
- Lawrence A. David and Eric J. Alm. (2011). Rapid evolutionary innovation during an Archaean genetic expansion. *Nature*, 469(7328):93-96.
- A tutorial on 16S rRNA gene and its use in microbiome research: http://greengenes.lbl.gov/cgi-bin/JD_Tutorial/nph-Tutorial_2Main2.cgi
- Dariush Mozaffarian, Tao Hao, Eric B. Rimm, Walter C. Willett, and Frank B. Hu. (2011). Changes in diet and lifestyle and long-term weight gain in women and men. *The New England journal of medicine*, 364(25):2392-2404.
- JH Hehemann, G Correc, T Barbeyron, W Helbert, M Czjzek, and G Michel. (2010). Transfer of carbohydrate- active enzymes from marine bacteria to japanese gut microbiota. *Nature*, 464(5):908-12.
- The Human Microbiome Jumpstart Reference Strains Consortium. (2010). A Catalog of Reference Genomes from the Human Microbiome. *Science*, 328(5981):994-999

6.11 Tools and techniques

6.12 What have we learned?

In this lecture, we learned about the field of bacterial genomics in general and how bacterial ecosystems can be used to verify major environmental changes at early stages of evolution (Study 1), can act as a noninvasive diagnostic tool (Study 2), are temporarily or permanently affected by different environmental and dietary factors (Study 3), can act as the link between diet and phenotype (Study 4), can cause antibiotic resistance genes to be carried between different species' microbiome through horizontal gene transfer (Study 5), and can be used to identify significant virulence factors in disease states (Study 6).

Bibliography

[1] The Human Microbiome Jumpstart Reference Strains Consortium. A Catalog of Reference Genomes from the Human Microbiome. *Science*, 328(5981):994–999, May 2010.

[2] Lawrence A. David and Eric J. Alm. Rapid evolutionary innovation during an Archaean genetic expansion. *Nature*, 469(7328):93–96, January 2011.

[3] JH Hehemann, G Correc, T Barbeyron, W Helbert, M Czjzek, and G Michel. Transfer of carbohydrate-active enzymes from marine bacteria to japanese gut microbiota. *Nature*, 464(5):908–12, 2010 Apr 8.

[4] Dariush Mozaffarian, Tao Hao, Eric B. Rimm, Walter C. Willett, and Frank B. Hu. Changes in diet and lifestyle and long-term weight gain in women and men. *The New England journal of medicine*, 364(25):2392–2404, June 2011.

Part II

Coding and Non-Coding Genes

CHAPTER

SEVEN

HIDDEN MARKOV MODELS I

Sumaiya Nazeen (Sep 25, 2012)
Chrisantha Perera (Sep 27, 2011)
Gleb Kuznetsov, Sheida Nabavi (Sep 28, 2010)
Elham Azizi (Sep 29, 2009)

Figures

7.1 Introduction

Hidden Markov Models (HMMs) are some of the most widely used methods in computational biology. They allow us to investigate questions such uncovering the underlying model behind certain DNA sequences. By representing data in rich probabilistic ways, we can ascribe meaning to sequences and make progress in endeavors including, but not limited to, Gene Finding.

This lecture is the first of two on HMMs. It covers Evaluation and Parsing. The next lecture on HMM's will cover Posterior Decoding and Learning with an eventual lead into the Expectation Maximization (EM) algorithm. We will eventually cover both supervised and unsupervised learning. In supervised learning, we have training data available that labels sequences with particular models. In unsupervised learning, we do not have labels so we must seek to partition the data into discrete categories based on discovered probabilistic similarities.

We find many parallels between HMMs and sequence alignment. Dynamic programming lies at the core of both methods, as solutions to problems can be viewed as being composed of optimal solutions to sub-problems.

7.2 Modeling

7.2.1 We have a new sequence of DNA, now what?

1. **Align it:**
   - with things we know about (database search).
   - with unknown things (assemble/clustering)
2. **Visualize it:** *"Genomics rule #1": Look at your data!*
   - Look for nonstandard nucleotide compositions.
   - Look for k-mer frequencies that are associated with protein coding regions, recurrent data, high GC content, etc.
   - Look for motifs, evolutionary signatures.
   - Translate and look for open reading frames, stop codons, etc.
   - Look for patterns, then develop machine learning tools to determine reasonable probabilistic models. For example by looking at a number of quadruples we decide to color code them to see where they most frequently occur.
3. **Model it:**
   - Make hypothesis.
   - Build a generative model to describe the hypothesis.
   - Use that model to find sequences of similar type.

   We're not looking for sequences that necessarily have common ancestors, rather, we're interested in similar properties. We actually don't know how to model whole genomes, but we can model small aspects of genomes. The task requires understanding all the properties of genome regions and computationally building generative models to represent hypotheses. For a given sequence, we want to annotate regions whether they are introns, exons, intergenic, promoter, or otherwise classifiable regions.

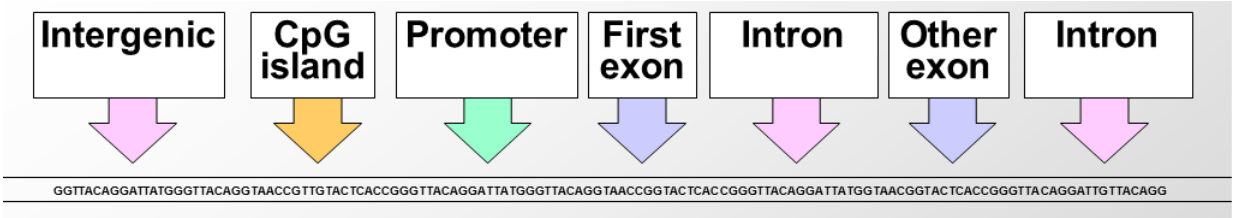


Figure 7.1: Modeling biological sequences

Building this framework will give us the ability to:

- Generate (emit) sequences of similar type according to the generative model
- Recognize the hidden state that has most likely generated the observation
- Learn (train) large datasets and apply to both previously labeled data (supervised learning) and unlabeled data (unsupervised learning).

In this lecture we discuss algorithms for emission and recognition.

7.2.2 Why probabilistic sequence modeling?

- Biological data is noisy.
- Probability provides a calculus for manipulating models.
- Not limited to yes/no answers, can provide degrees of belief.
- Many common computational tools are based on probabilistic models.
- Our tools: Markov Chains and HMM.

7.3 Motivating Example: The Dishonest Casino

7.3.1 The Scenario

Imagine the following scenario: You enter a casino that offers a dice-rolling game. You bet $1 and then you and a dealer both roll a die. If you roll a higher number you win $2. Now there's a twist to this seemingly simple game. You are aware that the casino has two types of dice:

1. Fair die: $P(1) = P(2) = P(3) = P(4) = P(5) = P(6) = 1/6$
2. Loaded die: $P(1) = P(2) = P(3) = P(4) = P(5) = 1/10$ and $P(6) = 1/2$

The dealer can switch between these two dice at any time without you knowing it. The only information that you have are the rolls that you observe. We can represent the state of the casino die with a simple Markov model:

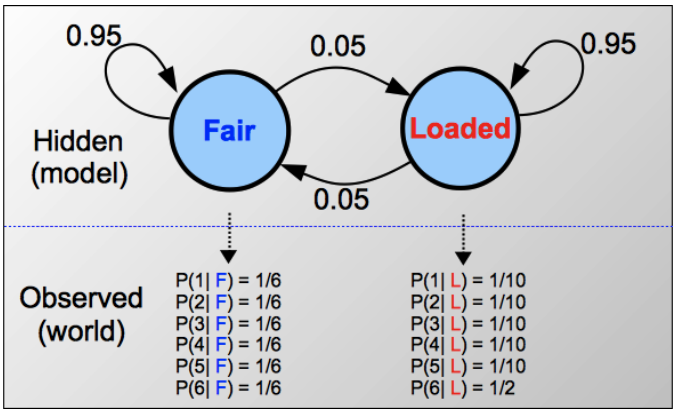


Figure 7.2: State of a casino die represented by a Hidden Markov model

The model shows the two possible states, their emissions, and probabilities for transition between them. The transition probabilities are educated guesses at best. We assume that switching between the states doesn't happen too frequently, hence the .95 chance of staying in the same state with every roll.

7.3.2 Staying in touch with biology: An analogy

For comparison, the figure below gives a similar model for a situation in biology where a sequence of DNA has two potential sources: injection by a virus versus normal production by the organism itself:

Given this model as a hypothesis, we would observe the frequencies of C and G to give us clues as to the source of the sequence in question. This model assumes that viral inserts will have higher CpG prevalence, which leads to the higher probabilities of C and G occurrence.

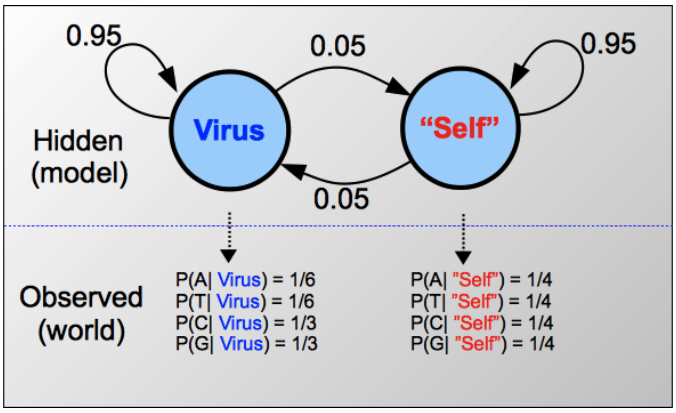


Figure 7.3: Potential DNA sources: viral injection vs. normal production

7.3.3 Running the Model

Let's look at a particular sequence of rolls and pick for the underlying model the two extreme cases: One in which the dealer is always using a fair die, and the other in which the dealer is always using a loaded die. We run the model on each to understand the implications. We'll build up to introducing HMMs by first trying to understand the likelihood calculations from the standpoint of fundamental probability principles of independence and conditioning.

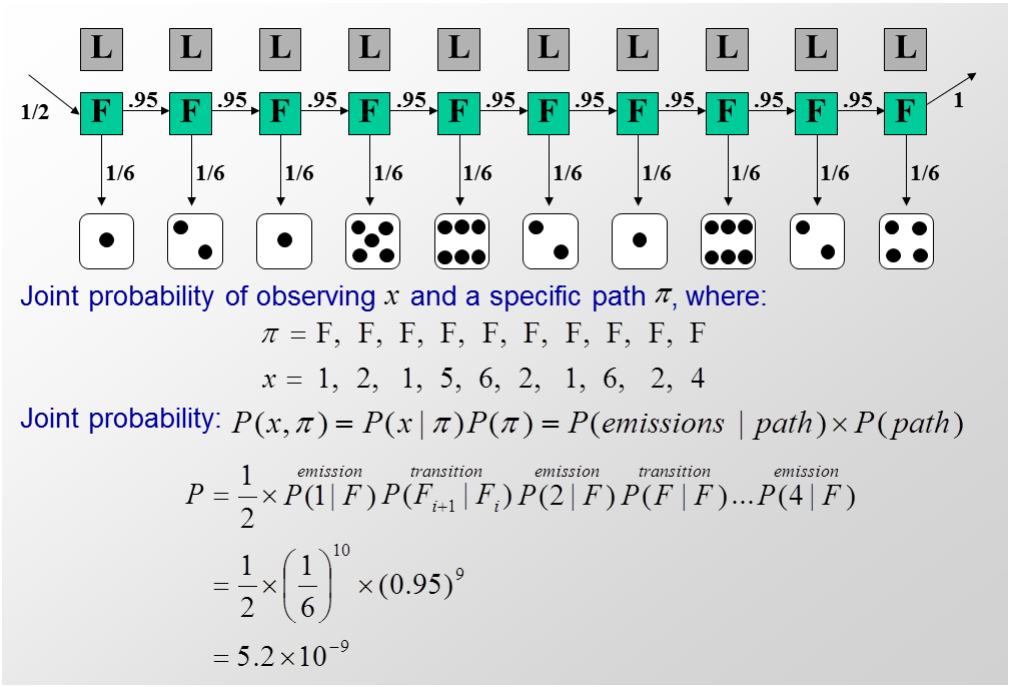


Figure 7.4: Running the model: probability of a sequence, given path consists of all fair dice

In the first case, where we assume the dealer is always using a fair die, we calculate the probability as shown in Figure 7.4. The product term has three components: 1/2 - the probability of starting with the fair die model, $(1/6)^{10}$ - the probability of the roll given the fair model (6 possibilities with equal chance), and lastly $(0.95)^9$ - the transition probabilities that keep us always in the same state.

The opposite extreme, where the dealer always uses a loaded die, has a similar calculation, except that we note a difference in the emission component. This time, 8 of the 10 rolls carry a probability of 1/10

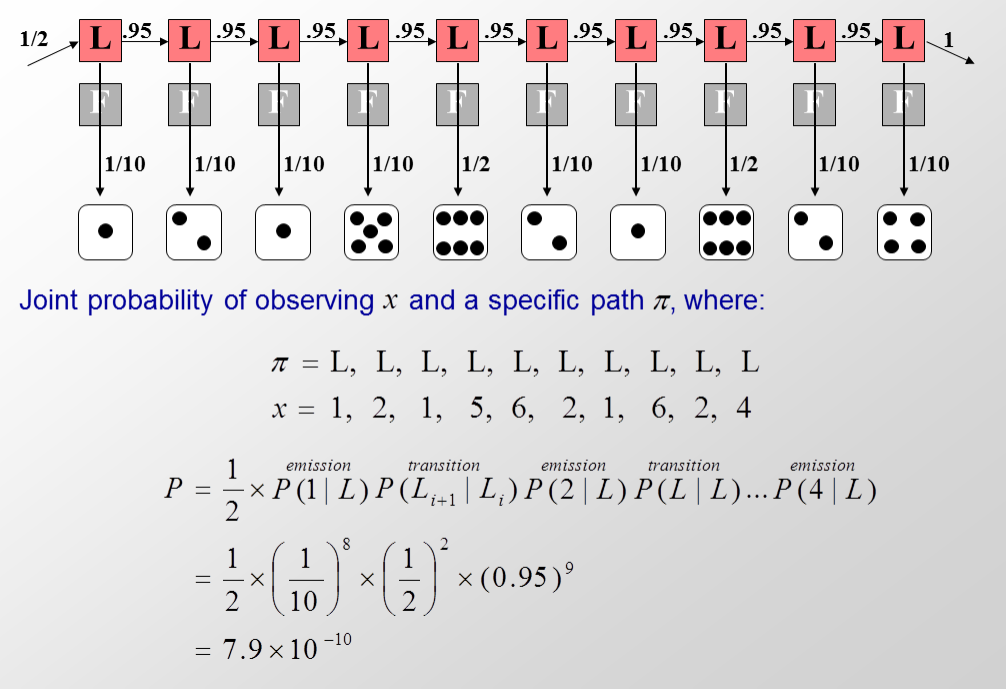


Figure 7.5: Running the model: probability of a sequence, given path consists of all loaded dice

based on the assumption of a loaded die underlying model, whiletwo rolls of 6 have a probability of 1/2 of occurring. Again we multiply all of these probabilities together according to principles of independence and conditioning. This is shown in Figure 7.5.

Finally, in order to make sense of the likelihoods, we compare them by calculating a likelihood ratio, as shown in Figure 7.6.

We find that, it is significantly more likely for the sequence to result from a fair die. Now we step back and ask, does this make sense? Well, of course. Two occurrences of rolling 6 in ten rolls doesn't seem out of the ordinary, so our intuition matches the results of running the model.

7.3.4 Adding Complexity

Now imagine the more complex, and interesting, case where the dealer switches the die at some point during the sequence. We make a guess at an underlying model based on this premise:

Again, we can calculate the likelihood using fundamental probabilistic methods. From Figure 7.7, we can see that, our guessed path is far less likely than the previous two cases. Now, it begins to dawn on us that we will need more rigorous techniques for inferring the underlying model. In the above cases we more-or-less just guessed at the model, but what we want is a way to systematically derive likely models. Let's formalize the models introduced thus far as we continue toward understanding HMM-related techniques.

7.4 Formalizing Markov Chains and HMMS

7.4.1 Markov Chains

A Markov Chain reduces a problem space to a finite set of states and the transition probabilities between them. At every time step, we observe the state we are in and simulate a transition, independent of how we got that state. More formally, a Markov Chain consists of:

- A set of states, Q.

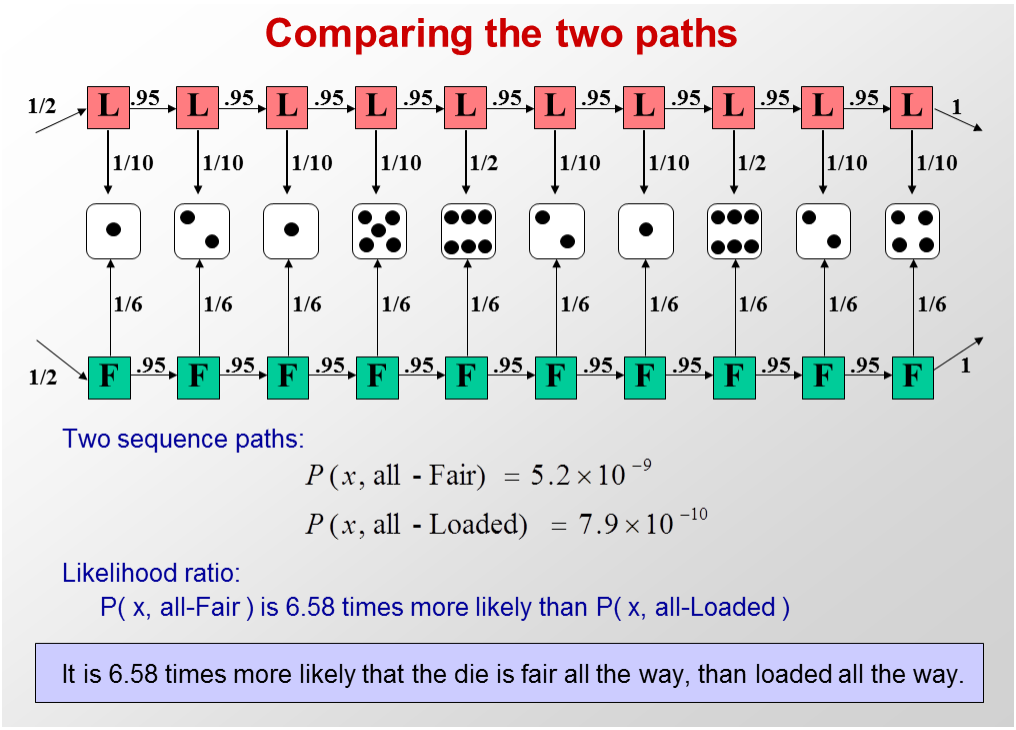


Figure 7.6: Comparing the two paths: all-Fair vs. all-loaded.

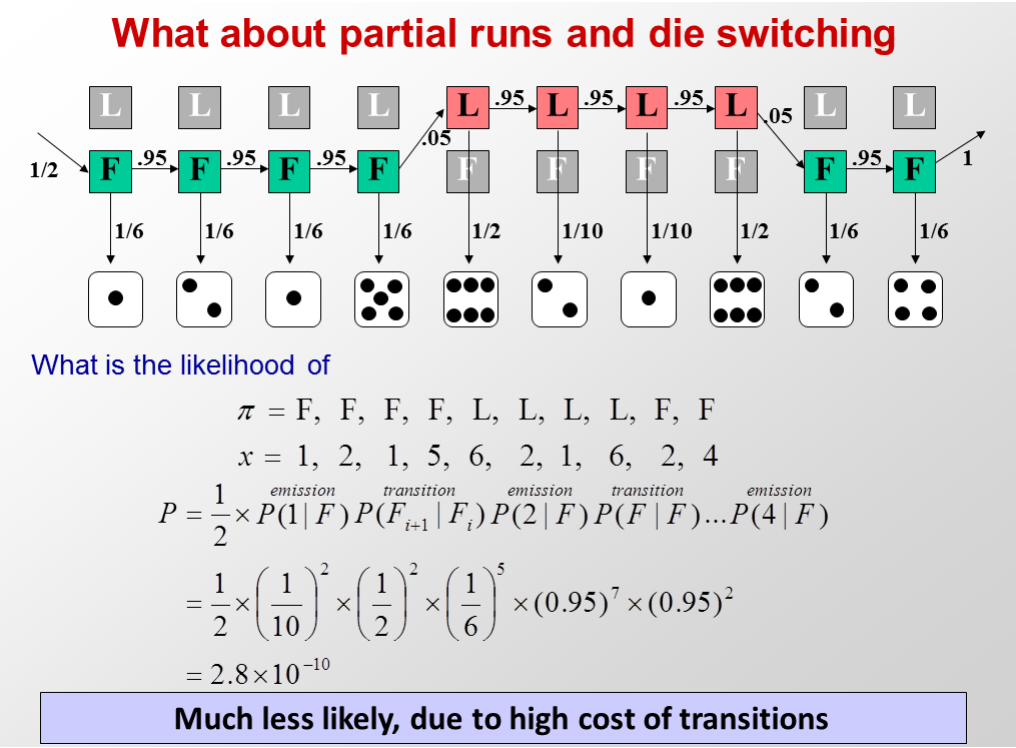


Figure 7.7: Partial runs and die switching

- A transition matrix, A whose elements correspond to the probabilities of transition from state i to state j.

- A vector of initial state probabilities , p.

The key property of Markov Chains is that they are memory-less, i.e., each state depends only on the previous state. So we can immediately define a probability for the next state, given the current state:

$$P(x_i|x_{i-1}, ..., x_1) = P(x_i|x_{i-1})$$

Therefore, the columns of A have to sum up to 1. In this way, the probability of the sequence can be decomposed into:

$$P(x) = P(x_L, x_{L-1}, ..., x_1) = P(x_L|x_{L-1})P(x_{L-1}|x_{L-2})...P(x_2|x_1)P(x_1)$$

$P(x_1)$ can also be calculated from the transition probabilities: If we multiply the initial state probabilities at time $t = 0$ by the transition matrix, we get the probabilities of states at time $t = 1$ and therefore we also have them for time $t = n$.

7.4.2 Hidden Markov Models

Hidden Markov Models are used as a representation of a problem space in which observations come about as a result of states of a system which we are unable to observe directly. These observations, or emissions, result from a particular state based on a set of probabilities. Thus HMMs are Markov Models where the states are hidden from the observer and instead we have observations generated with certain probabilities associated with each state. These probabilities of observations are known as emission probabilities.

Formally, a Hidden Markov Model consists of the following parameters:

- A series of states, Q.
- A transition matrix, A: For each s, t, in Q the transition probability is: $a_{st} = P(x_i = t|x_{i-1} = s)$
- A vector of initial state probabilities , p.
- Set of observation symbols, V, for example {A, T, C, G} or, 20 amino-acids or utterances in human language.
- A matrix of emission probabilities, E: For each s, t, in Q, the emission probability is

$$e_{sk} = P(v_k \text{ at time } t|q_t = s)$$

The key property of memory-less-ness is inherited from Markov Models: the emissions and transitions are only dependent on the current state and not on the past history.

7.5 Back to Biology

Now that we have formalized HMMs, we want to use them HMMs in solving some real biological problems. In fact, HMMs are a great tool for gene sequence analysis, because we can look at a sequence of DNA as being emitted by a mixture of models. These may include introns, exons, transcription factors, etc. While we may have some sample data that matches models to DNA sequences, in the case that we start fresh with a new piece of DNA, we can use HMMs to ascribe some potential models to the DNA in question. We will first introduce a simple example and think about it a bit. Then, we will discuss some applications of HMM in solving interesting biological questions, before finally describing the HMM techniques that solve the problems that arise in such a first-attempt/native analysis.

7.5.1 A simple example: Finding GC-rich regions

Imagine the following scenario: we are trying to find GC rich regions by modeling nucleotide sequences drawn from two different distributions: background and promoter. Background regions have uniform distribution of 0.25 for each of A, T, G, C. Promoter regions have probabilities: A: 0.15, T: 0.13, G: 0.30, C: 0.42. Given one nucleotide observed, we cannot say anything about the region from which it was originated, because both regions could have emited it at different probabilities. We can learn these initial state probabilities based in steady state probabilities. By looking at a sequence, we want to identify which regions originate from a background distribution (B) and which regions are from a promoter model (P). It was noted again that Markov chains are absolutely memory-less, this means that for example if you have lost in a Casino in the last 7 days, it won't mean that you would most probably win today. This is also true in the example of rolling a die where you might have repetitions of a number.

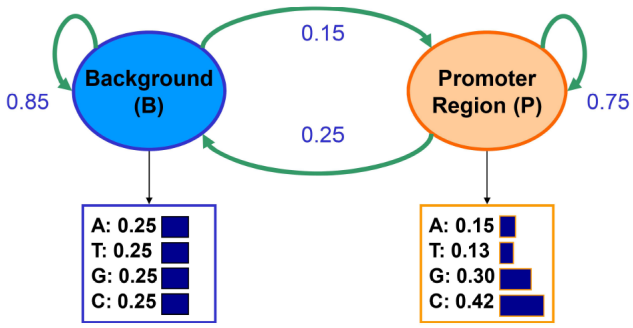


Figure 7.8: HMMS as a generative model for finding GC-rich regions.

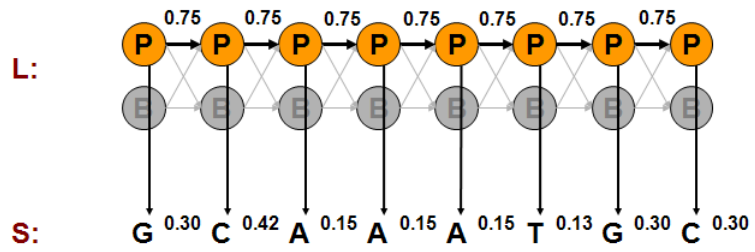
We are given the transition and emission probabilities based on relevant abundance and average length of regions where x = vector of observable emissions consisting of symbols from the alphabet {A,T,G,C}; π = vector of states in a path (e.g. BPPBP); π^* = maximum likelihood of generating that path. In our interpretation of sequence, the max likelihood path will be found by incorporating all emissions transition probabilities (by dynamic programming).

HMMs are generative models, in that an HMM gives the probability of emission given a state (with Bayes' Rule), essentially telling you how likely the state is to generate those sequences. So we can always run a generative model for transition between states and start anywhere. In Markov Chains, the next state will give different outcomes with different probabilities. No matter which the next state is, at that next state, the next symbol will still come out with different probabilities. HMMs are similar: You can pick an initial state based on initial probability vector. In the example above, you pick B since there are more background than promoter. Then draw an emission from the P(X—B). Since all are 0.25, you can pick any symbol, for example G. Given this emission, the probability of the next transition state does not change. So we have transition to B again is with probability 0.85 and to P with 0.15 so we go with B and so on.

We can compute the probability of one such generation by multiplying the probabilities that the model makes exactly the choices we assumed. For the example above we have the probability of three different paths computed as in Figures 7.9, 7.10, and 7.11.

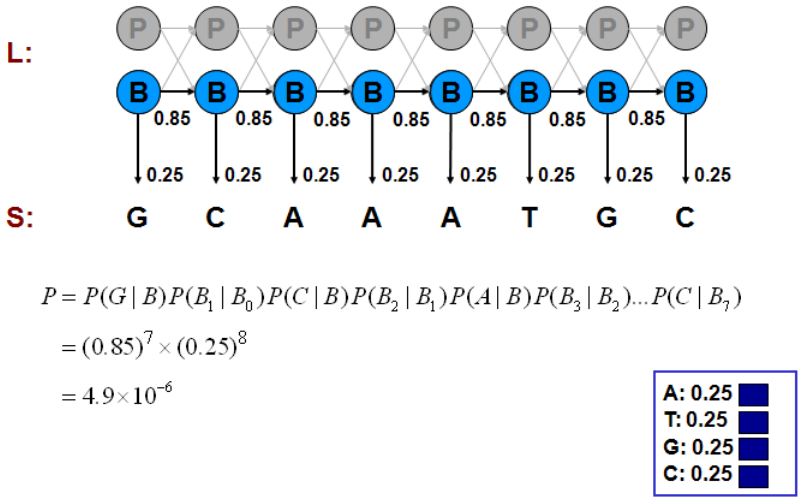
Now the question is, what path is most likely to generate the given sequence?

One brute force approach may be looking at all paths, trying all possibilities, and calculating their joint probability $P(x, \pi)$. The sum of probabilities of all the alternatives is 1. For example, if all states are promoters, $P(x, \pi) = 9.3 \times 10^{-7}$. If all emissions are Gs, $P(x, \pi) = 4.9 \times 10^{-6}$. If we have use the mixture of B's and P's as in Figure 7.11, $P(x, \pi) = 6.7 \times 10^{-7}$; which is small because a lot of penalty is paid for the transitions between B's and P's which are exponential in length of sequence. Usually, if you observe more G's, it is more likely to be in the promoter region and if you observe more A and Ts, then it is more likely to be in the background. But we need something more than just observation to support our belief. In the coming sections we will see how can we mathematically support our intuition.



$$P(x,\pi)=a_P*e_P(G)*a_{PP}*e_P(G)*a_{PP}*e_P(C)*a_{PP}*e_P(A)*a_{PP}*\dots$$
$$=a_p*(0.75)^7*(0.15)^3*(0.13)^1*(0.30)^2*(0.42)^2$$
$$=9.3*10^{-7}$$

Figure 7.9: Probability of seq, path if all promoter



$$P = P(G \mid B)P(B_1 \mid B_0)P(C \mid B)P(B_2 \mid B_1)P(A \mid B)P(B_3 \mid B_2)\dots P(C \mid B_7)$$
$$= (0.85)^7 \times (0.25)^8$$
$$= 4.9 \times 10^{-6}$$

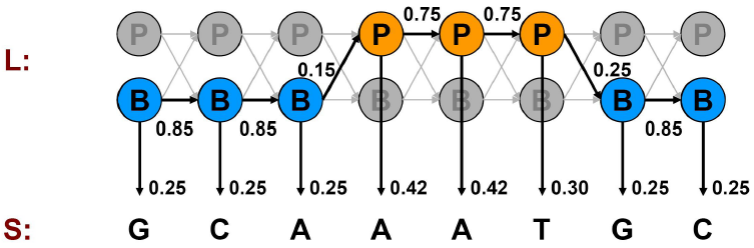
Figure 7.10: Probability of seq, path if all background

7.5.2 Application of HMMs in Biology

HMMs are used in answering many interesting questions. Some biological application of HMMs are summarized in the table shown in Figure 7.12.

7.6 Algorithmic Settings for HMMs

We use HMMs for three types of operation: **scoring**, **decoding**, and **learning**. We talk about scoring and decoding in this lecture. These operations can happen for a single path or all possible paths. For the single path operations, our focus is on discovering the path with maximum probability, the most likely path. However in all paths operations we are interested in a sequence of observations or emissions regardless of its corresponding paths.



$$P = P(G \mid B)P(B_1 \mid B_0)P(C \mid B)P(B_2 \mid B_1)P(A \mid B)P(P_3 \mid B_2)...P(C \mid B_7)$$
$$= (0.85)^3 \times (0.25)^6 \times (0.75)^2 \times (0.42)^2 \times 0.30 \times 0.15$$
$$= 6.7 \times 10^{-7}$$

Figure 7.11: Probability of seq, path sequence if mixed

Application	Detection of GC-rich region	Detection of Conserved region	Detection of Protein coding exons	Detection of Protein coding conservation	Detection of Protein coding gene structures	Detection of chromatin states
Topology / Transitions	2 states, different nucleotide composition	2 states, different conservation levels	2 states, different tri-nucleotide composition	2 states, different evolutionary signatures	~20 states, different composition / conservation, specific structure	40 states, different chromatin mark combinations
Hidden States / Annotation	GC-rich / AT-rich	Conserved / non-Conserved	Coding (exon) / non-Coding (intron or intergenic)	Coding (exon) / non-Coding (intron or intergenic)	First / last / middle coding exon, UTRs, intron 1/2/3, intergenic, *(+,-) strand	Enhancer / Promoter / Transcribed / Repressed / Repetitive
Emissions / Observations	Nucleotides	Level of conservation	Triplets of nucleotides	64 x 64 matrix of codon substitution frequencies	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies

Figure 7.12: Some biological applications of HMM

7.6.1 Scoring

Scoring over a single path

For a single path we define the Scoring problem as follows:

- **Input:** A sequence of observations $x = x_1 x_2 \dots x_n$ generated by an HMM $M(Q, A, p, V, E)$ and a path of states $\pi = \pi_1 \pi_2 \dots \pi_n$.
- **Output:** Joint probability, $P(x, \pi)$ of observing x if the hidden state sequence is π.

The single path calculation is essentially the likelihood of observing the given sequence over a particular path using the following formula:

$$P(x, \pi) = P(x|\pi)P(\pi)$$

We have already seen the examples of single path scoring in our Dishonest Casino and GC-rich region examples.

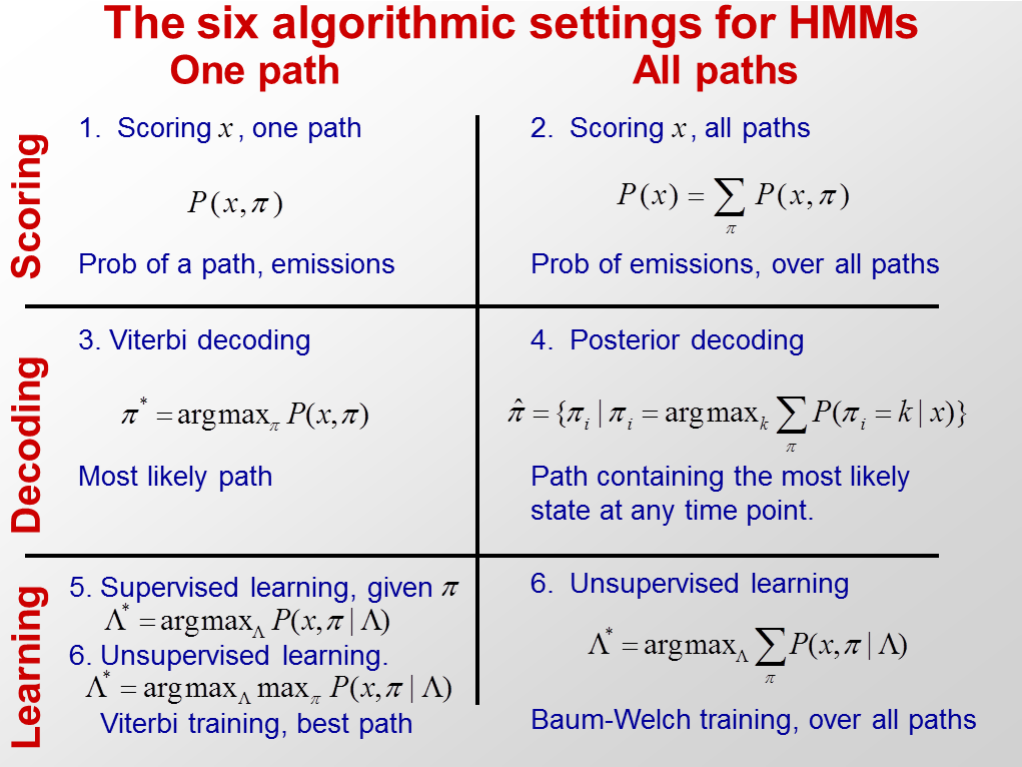


Figure 7.13: The six algorithmic settings for HMMS

Scoring over all paths

We define the all paths version of Scoring problem as follows:

- **Input:** A sequence of observations $x = x_1 x_2 \dots x_n$ generated by an HMM $M(Q, A, p, V, E)$.
- **Output:** Joint probability, $P(x, \pi)$ of observing x over all possible hidden state sequences, πs.

we do all operations on all possible paths and states and score the sequence over all paths π using the following formula,

$$P(x) = \Sigma_{\pi} P(x, \pi)$$

In this case the score is calculated for just a given sequence of observations or emissions regardless of the paths. We use this score when we are interested in knowing the likelihood of particular sequence for a given HMM.

7.6.2 Decoding

Decoding answers the question: Given some observed sequence, what path gives us the maximum likelihood of observing this sequence? Formally we define the problem as follows:

- **Decoding over a single path:**
  - Input: A sequence of observations $x = x_1 x_2 \dots x_N$ generated by an HMM $M(Q, A, p, V, E)$.
  - Output: The most probable path of states, $\pi^* = \pi_1^* \pi_2^* \dots \pi_N^*$
- **Decoding over all paths:**
  - Input: A sequence of observations $x = x_1 x_2 \dots x_N$ generated by an HMM $M(Q, A, p, V, E)$.

– Output: The path of states, $\pi^* = \pi_1^*\pi_2^* \dots \pi_N^*$ that contains the most likely states at any time point.

In this lecture, we will look only at the problem of decoding over a single path. The problem of decoding over all paths will be discussed in the next lecture.

For the single path decoding problem, we can imagine a brute force approach where we calculate the joint probabilities of a given emission sequence and all possible paths and then pick the path with the maximum joint probability. The problem is - there are exponential number of paths and using such a brute force search for the maximum likelihood among all the paths is very time consuming and impractical. To solve this problem **Dynamic Programming** can be used. Let us formulate the problem in the dynamic programming approach.

Through decoding we would like to find out the most likely sequence of states based on the observation. For the decoding operation, the model parameters $e_i s$ (the emission probabilities given their states) and $a_{ij} s$, (the state transition probabilities) are given. Also, the sequence of emissions x is given. The goal is to find the sequence of hidden states, π^*, which maximizes the joint probability of a given emission and its path $P(x, \pi)$, i.e.,

$$\pi^* = \arg\max_{\pi} P(x, \pi)$$

Given the emitted sequence x we can evaluate any path through hidden states. However we are looking for the best path. We start by looking for the optimal substructure of this problem.

For a best path, we can say that, the best path through a given state must contain within it the following:

- Best path to previous state
- Best transition from previous state to this state
- Best path to the end state

Therefore the best path can be obtained based on the best path of the previous states, i.e., we can find a recurrence for the best path. The **Viterbi** algorithm, a dynamic programming algorithm, is commonly used to obtain the best path.

Most probable state path: the Viterbi algorithm

Suppose, the probability $v_k(i)$, of the most likely path ending at state k at position (or time instance) i in the path (which can be shown as $\pi_i = k$) is known for all the states k. Then we can compute this probability at time $i + 1$, as follows:

$$v_l(i + 1) = e_l(x_{i+1}) max_k(a_{kl} v_k(i))$$

The most probable path π^* or the maximum $P(x, \pi)$ can be found recursively. Assuming we know $v_j(i - 1)$, the score of the maximum path up to time $i - 1$, now we need to increase the computation for the next time step. The new maximum score path for each state depends on

- the maximum score of the previous states
- the penalty of transition to the new states (transition probability), and
- the emission probability.

In other words, the new maximum score for a particular state at time i is the one that maximizes the transition of all possible previous states to that particular state (the penalty of transition multiplied by their maximum previous scores multiplied by emission probability at the current time).

All sequences have to start in state 0 (the begin state). By keeping pointers backwards, the actual state sequence can be found by backtracking. The solution of this Dynamic Programming problem is very similar to the alignment algorithms that we studied in previous lectures.

The steps of the Viterbi algorithm **??** are summarized below:

1. Initialization ($i = 0$): $v_0(0) = 1$, $v_k(0) = 0$ for $k > 0$.

2. Recursion $(i = 1 \dots N)$: $v_l(i) = e_l(x_i) max_k(a_{kl} v_k(i-1))$; $ptr_i(l) = \arg\max_k(a_{kl} v_k(i-1))$.
3. Termination: $P(x, \pi^*) = max_k(v_k(N) a_{k0})$; $\pi^*_N = \arg\max_k(v_k(N) a_{k0})$.
4. Traceback $(i = N \dots 1)$: $\pi^*_{i-1} = ptr_i(\pi^*_i)$.

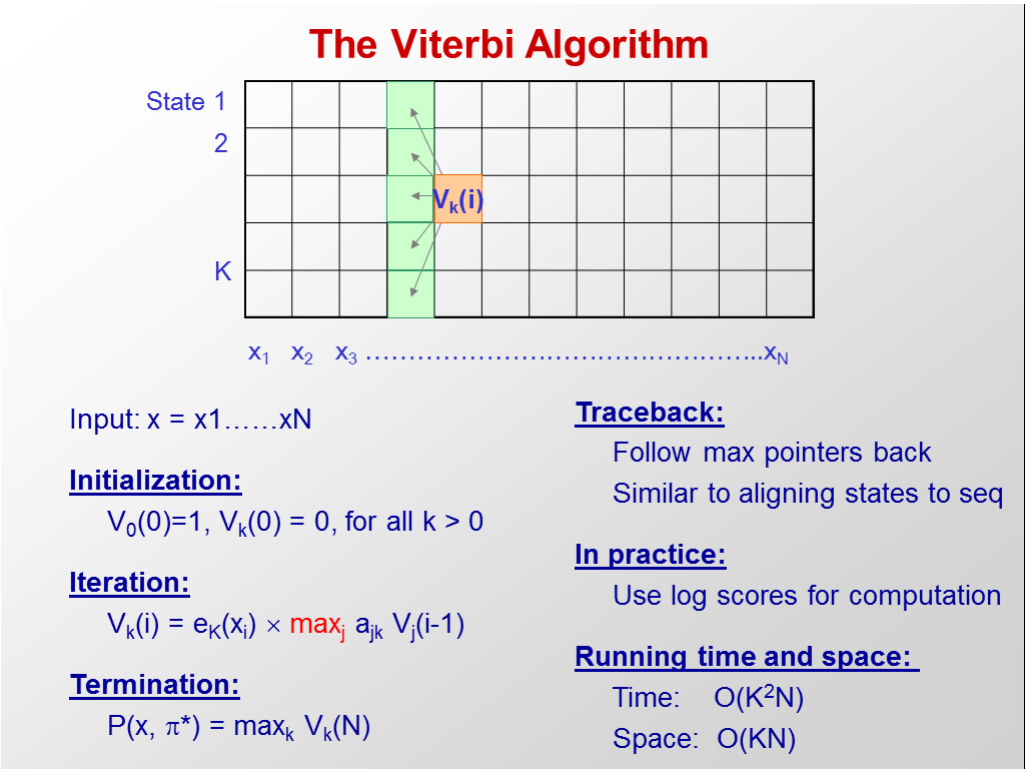


Figure 7.14: The Viterbi algorithm

As we can see in Figure 7.14, we fill the matrix from left to right and trace back. Each position in the matrix has K states to consider and there are KN cells in the matrix, so, the required computation time is $O(K^2N)$ and the required space is $O(KN)$ to remember the pointers. Note that, the running time has reduced from exponential to polynomial.

7.6.3 Evaluation

Evaluation is about answering the question: How well does our model of the world capture the actual world? Given a sequence x, many paths can generate this sequence. The question is how likely is the sequence, given the entire model? In other words, is this a good model? Or, how well the model does capture the exact characteristics of a particular sequence? We use evaluation operation of HMMs to answer these questions. With evaluation we can compare the models.

Let us formally define the Evaluation problem first.

- Input: A sequence of observations $x = x_1 x_2 \dots x_N$ and an HMM $M(Q, A, p, V, E)$.
- Output: The probability that x was generated by M summed over all paths.

We know that, if we are given an HMM, we can generate a sequence of length n easily using the following steps:

- Start at state π_1 according to probability $a_{0\pi_1}$ (obtained using vector, p).
- Emit letter x_1 according to emission probability $e_{\pi_1}(x_1)$.

- Go to state π_2 according to the transition probability $a_{\pi_1|\pi_2}$
- Keep doing this until emit x_N.

Thus we can emit any sequence and calculate its likelihood. However, many state sequence can emit the same x. Then, how do we calculate the total probability of generating a given x over all paths? That is, our goal is to obtain the following probability:

$$P(x|M) = P(x) = \sum_{\pi} P(x, \pi) = \sum_{\pi} P(x|\pi)P(\pi)$$

The challenge of obtaining this probability is there are too many paths (exponential number of paths). Each path has an associated probability. Some paths are more likely, some are less likely, and we need to sum them up all. One approach may be using just the Viterbi path and ignoring the others. We already have shown how to obtain the Viterbi path. But its probability is very small; it is just a small fraction of the probability mass of all possible paths. It is a good approximation only if it has high probability density. In other cases, it will give us an inaccurate approximation. So, the correct approach calculating the exact sum iteratively by using dynamic programming. The algorithm that does this is known as **Forward Algorithm**.

The Forward Algorithm

First we derive the formula for forward probability $f(i)$.

$$\begin{aligned}
f_l(i) &= P(x_1 \dots x_i, \pi = l) && (7.1)\\
&= \sum_{\pi_1 \dots \pi_{i-1}} P(x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-2}, \pi_{i-1}, \pi_i = l) e_l(x_i) && (7.2)\\
&= \sum_k \sum_{\pi_1 \dots \pi_{i-2}} P(x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-2}, \pi_{i-1} = k) a_{kl} e_l(x_i) && (7.3)\\
&= \sum_k f_k(i-1) a_{kl} e_l(x_i) && (7.4)\\
&= e_l(x_i) \sum_k f_k(i-1) a_{kl} && (7.5)\\
& && (7.6)
\end{aligned}$$

The full algorithm[2] is summarized below:

- Initialization ($i = 0$): $f_0(0) = 1$, $f_k(0) = 0$ for $k > 0$.
- Recursion ($i = 1 \dots N$): $f_l(i) == e_l(x_i) \sum_k f_k(i-1) a_{kl}$.
- Termination: $P(x) == \sum_k f_k(N) a_{k0}$

From Figure 7.15, it can be seen that, the Forward algorithm is very similar to the Viterbi algorithm. In the Forward algorithm, summation is used instead of maximization. Here we can reuse computations of the previous problem including penalty of emissions, penalty of transitions and sums of previous states. The required computation time is $O(K^2N)$ and the required space is $O(KN)$. The drawback of this algorithm is that in practice taking the sum of logs is difficult; therefore approximations and scaling of probabilities are used instead.

7.7 An Interesting Question: Can We Incorporate Memory in Our Model?

The answer to this question is - Yes, we can! But how? Recall that, Markov models are memoryless. In other words, all memory of the model is enclosed in states. So, the thing that we can do to store additional

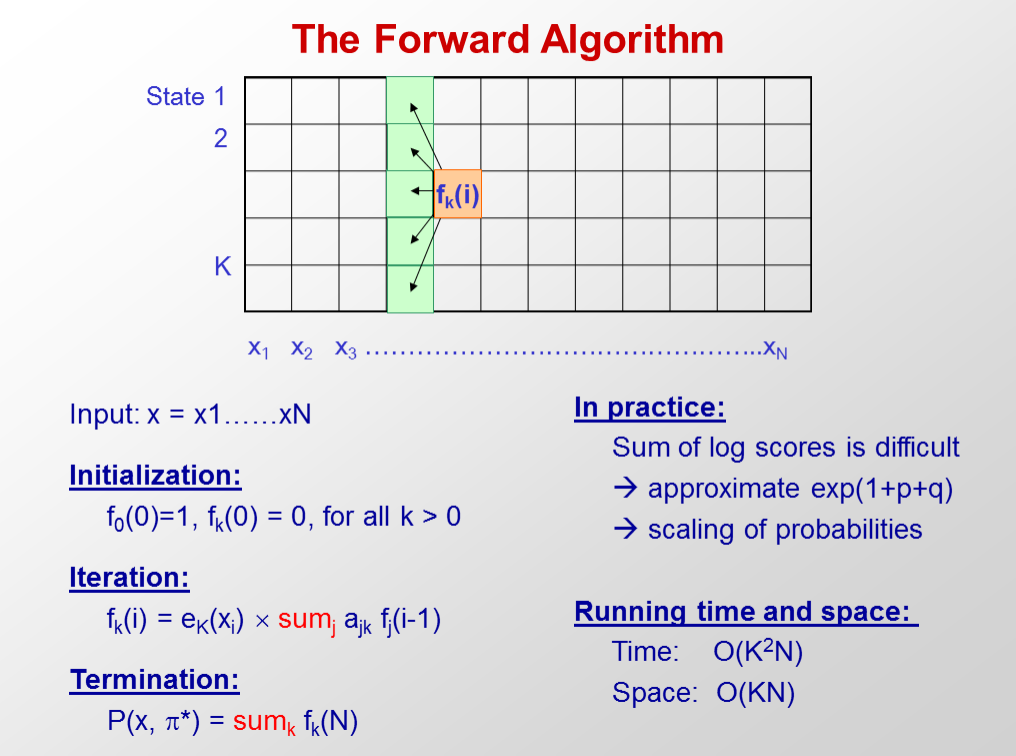


Figure 7.15: The Forward algorithm

information is to augment the number of states. Now, look back to the biological example we gave in Section 7.5.1. In our model, state emissions were dependent only on current state. And, the current state encoded only one nucleotide. But, what if we want our model to count di-nucleotide frequencies (for CpG islands[1]), or, tri-nucleotide frequencies (for codons), or di-codon frequencies involving six-nucleotide? We need to expand number of states. This is illustrated in Figures 7.16 and .7.17.

Here we show an HMM for CpG islands. Recall that, in our first biological example we used only two states - Background (non GC-rich) and Promoter (GC-rich) region, but here we have eight states to remember what nucleotide we have seen before, so that we can count the di-nucleotide frequencies. More discussion on this will follow in the next lecture.

7.8 Further Reading

7.8.1 Length Distributions of States and Generalized Hidden Markov Models

Given a Markov chain with the transition from any state to the end state having probability τ, the probability of generating a sequence of length L (and then finishing with a transition to the end state) is given by:

$$\tau(1-\tau)^{L-1}$$

Similarly, in the HMMs that we have been examining, the length of states will be exponentially distributed, which is not appropriate for many purposes. (For example, in a genomic sequence, an exponential distribution does not accurately capture the lengths of genes, exons, introns, etc). How can we construct a model that does not output state sequences with an exponential distribution of lengths? Suppose we want to make sure that our sequence has length exactly 5? We might construct a sequence of five states with only a single path permitted by transition probabilities. If we include a self loop in one of the states, we will

[1]CpG stands for C-phosphate-G. So, CpG island refers to a region where GC di-nucleotide appear on the same strand.

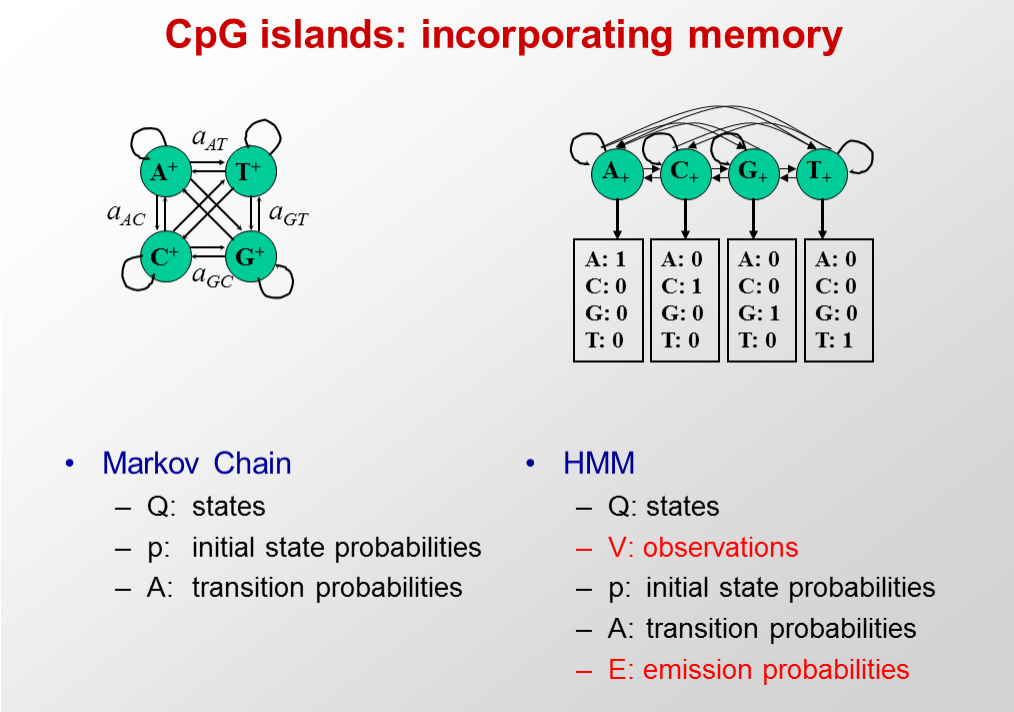


Figure 7.16: CpG Islands - Incorporating Memory

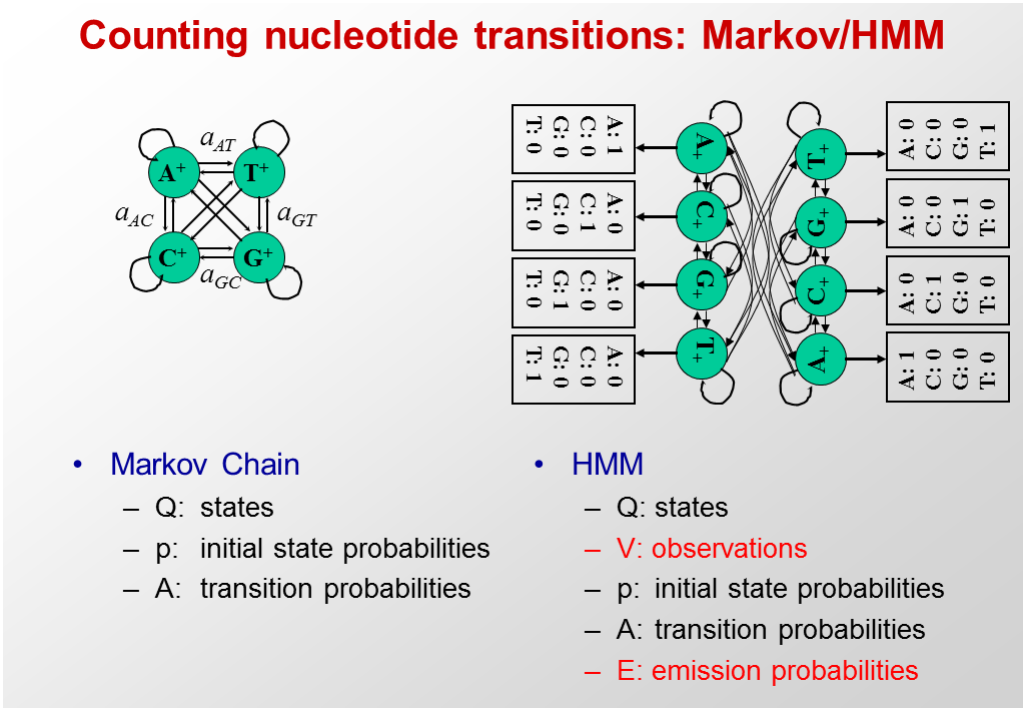


Figure 7.17: Counting Nucleotide Transitions

output sequences of minimum length 5, with longer sequences exponentially distributed. Suppose we have a chain of n states, with all chains starting with state π_1 and transitioning to an end state after π_n. Also assume that the transition probability between state π_i and π_{i+1} is $1-p$, while the self transition probability

of state π_i is p. The probability that a sequence generated by this Markov chain has length L is given by:

$$\binom{L-1}{n-1} p^{L-n}(1-p)^n$$

This is called the negative binomial distribution.

More generally, we can adapt HMMs to produce output sequences of arbitrary length. In a *Generalized Hidden Markov Model* [1] (also known as a *hidden semi-Markov model*), the output of each state is a string of symbols, rather than an individual symbol. The length as well as content of this output string can be chosen based on a probability distribution. Many gene finding tools are based on generalized hidden Markov models.

7.8.2 Conditional random fields

Conditional random field model is an alternative to HMMs. It is a discriminative undirected probabilistic graphical model. It is used to encode known relationships between observations and construct consistent interpretations. It is often used for labeling or parsing of sequential data. It is widely used in gene finding. The following resources can be helpful in order to learn more about CRFs:

- Lecture on Conditional Random Fields from Probabilistic Graphical Models course: https://class.coursera.org/pgm/lecture/preview/33. For background, you might also want to watch the two previous segments, on pairwise Markov networks and general Gibbs distributions.
- Conditional random fields in biology: http://www.cis.upenn.edu/~pereira/papers/crf.pdf
- Conditional Random Fields tutorial: http://people.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf

7.9 Current Research Directions

7.10 Tools and Techniques

7.11 What Have We Learned?

Bibliography

[1] Introduction to GHMMs: www.cs.tau.ac.il/~rshamir/algmb/00/scribe00/html/lec07/node28.html.

[2] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. eleventh edition, 2006.

CHAPTER

EIGHT

HIDDEN MARKOV MODELS II - POSTERIOR DECODING AND LEARNING

Charalampos Mavroforakis and Chidube Ezeozue (2012)
Thomas Willems (2011)
Amer Fejzic (2010)
Elham Azizi (2009)

Figures

8.1 Review of previous lecture

8.1.1 Introduction to Hidden Markov Models

In the last lecture, we familiarized ourselves with the concept of discrete-time Markov chains and Hidden Markov Models (HMMs). In particular, a Markov chain is a discrete random process that abides by the Markov property, i.e. that the probability of the next state depends only on the current state. To model how states change from step to step, the Markov chain uses a matrix of transition probabilities. In addition, it is characterized by a one-to-one correspondence between the states and observed symbols. More formally, a Markov chain is fully defined by the following variables:

- $\pi_i \in Q$, the state at the i^{th} step in a sequence of finite states Q of length N that can hold a value from a finite alphabet Σ of length K
- a_{jk}, the transition probability of moving from state j to state k, $P(\pi_i = k|\pi_{i-1} = j)$, for each j, k in Q
- $a_{0j} \in P$, the probability that the initial state will be j

Examples of Markov chains are abundant in everyday life. In the last lecture, we considered the canonical example of a weather system in which each state is either rain, snow, sun or clouds and the observables of the system correspond exactly to the underlying state. Suppose, however, that the weather results from some unknown underlying seasonal states. In this situation, the states are considered "hidden" and no longer

share a one-to-one correspondence with the observables. These types of situations require a generalization of Markov chains known as Hidden Markov Models (HMMs).

Did You Know?
Markov Chains may be thought of as WYSIWYG - What You See Is What You Get

HMMs incorporate additional elements to model the disconnect between the observables of a system and the hidden states. For a sequence of length N, each observable state is instead replaced by a hidden state and a character emitted from that state. It is important to note that characters from each state are emitted according to a series of emission probabilities. More formally, the two additional descriptors of an HMM are:

- $x_i \in X$, the emission at the i^{th} step in a sequence of finite characters X of length N that can hold a character from a finite set of observation symbols $v_l \in V$
- $e_k(v_l) \in E$, the emission probability of emitting character v_l when the state is k, $P(x_i = v_l | \pi_i = k)$

In summary, an HMM is defined by the following variables:

- a_{jk}, $e_k(v_l)$, and a_{0j} that model the discrete random process
- π_i, the sequence of hidden states
- x_i, the sequence of observed emissions

8.1.2 Genomic Applications of HMMs

The figure below shows some genomic applications of HMMs

Application	Detection of GC-rich region	Detection of Conserved region	Detection of Protein coding exons	Detection of Protein coding conservation	Detection of Protein coding gene structures	Detection of chromatin states
Topology / Transitions	2 states, different nucleotide composition	2 states, difference conservation levels	2 states, different tri-nucleotide composition	2 states, different evolutionary signatures	~20 states, different composition / conservation, specific structure	40 states, different chromatin mark combinations
Hidden States / Annotation	GC-rich / AT-rich	Conserved/ non-Conserved	Coding (exon) / non-Coding (intron or intergenic)	Coding (exon) / non-Coding (intron or intergenic)	First / last / middle coding exon, UTRs, intron 1/2/3, intergenic, *(+,-) strand	Enhancer / Promoter / Transcribed / Repressed / Repetitive
Emissions / Observations	Nucleotides	Level of conservation	Triplets of nucleotides	64 x 64 matrix of codon substitution frequencies	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies

Figure 8.1: Genomic applications of HMMs

Niceties of some of the applications shown in figure 8.1 include:

- **Detection of Protein coding conservation**
  This is similar to the application of detecting protein coding exons because the emissions are also not nucleotides but different in the sense that, instead of emitting codons, substitution frequencies of the codons are emitted.

- **Detection of Protein coding gene structures**
  Here, it is important for different states to model first, last and middle exons because first exons go through a start codon and last exons go through a stop codon and that knowledge has to be encoded. This differs from the application of detecting protein coding exons because in the latter, the nature of the coding exons is unimportant.
  It is also important to differentiate between introns 1,2 and 3 so that the reading frame between one exon and the next exon can be remembered e.g. if one exon stops at the second codon position, the next one has to start at the third codon position. Therefore, the additional intron states encode the codon position.

- **Detection of chromatin states**
  Chromatin state models are dynamic and vary from cell type to cell type so every cell type will have its own annotation. They will be discussed in fuller detail in the genomics lecture including strategies for stacking/concatenating cell types.

8.1.3 Viterbi decoding

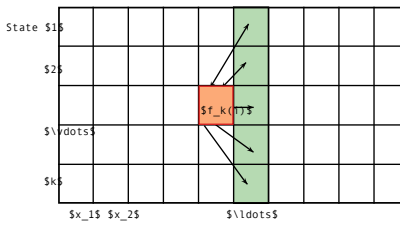
Previously, we demonstrated that when given a full HMM (Q, A, X, E, P), the likelihood that the discrete random process produced the provided series of hidden states and emissions is given by:

$$P(x_1, \ldots, x_N, \pi_1, \ldots, \pi_N) = a_{0\pi_i} \prod_i e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}} \tag{8.1}$$

This corresponds to the total joint probability, $P(x, \pi)$. Usually, however, the hidden states are not given and must be inferred. One solution to this *decoding* problem is known as the **Viterbi decoding** algorithm. Running in $O(K^2N)$ time and $O(KN)$ space, where K is the number of states, this algorithm determines the sequence of hidden states (the path π^*) that maximizes the joint probability of the observables and states, i.e. $P(x, \pi)$. Essentially, this algorithm defines $V_k(i)$ to be the probability of the most likely path through state $\pi_i = k$, and it recursively computes $V_k(i) = e_k(x_i) \times \max_j(V_j(i-1)a_{jk})$

8.1.4 Forward Algorithm

Another problem we touched was that, instead of computing the probability of a single path of hidden state emitting the observed sequence, we may want to calculate the probability of the sequence being produced by all possible paths. In order to do that, we proposed the **Forward algorithm**, which is described in Figure 8.2



Input: $x = x_1 \dots x_N$

Initialization:
$f_0(0) = 1, f_k(0) = 0, \quad$ for all $k > 0$

Iteration:
$f_k(i) = e_k(x_i) \times \sum_j a_{jk} f_j(i-1)$

Termination:
$P(x, \pi^*) = \sum_k f_k(N)$

Figure 8.2: The Forward Algorithm

The forward algorithm first calculates the joint probability of observing the first t emitted characters and being in state k at time t. More formally,

$$f_k(t) = P(\pi_t = k, x_1, \dots, x_t) \tag{8.2}$$

Given that the number of paths is exponential in t, dynamic programming must be employed to solve this problem. We can develop a simple recursion for the forward algorithm by employing the Markov property as follows:

$$f_k(t) = \sum_l P(x_1, \dots, x_t, \pi_t = k, \pi_{t-1} = l) = \sum_l P(x_1, \dots, x_{t-1}, \pi_{t-1} = l) * P(x_t, \pi_t | \pi_{t-1}) \tag{8.3}$$

Recognizing that the first term corresponds to $f_l(t-1)$ and that the second term can be expressed in terms of transition and emission probabilities, this leads to the final recursion:

$$f_k(t) = e_k(x_t) \sum_l f_l(t-1) * a_{lk} \tag{8.4}$$

Intuitively, one can understand this recursion as follows: Any path that is in state k at time t must have come from a path that was in state l at time $t-1$. The contribution of each of these sets of paths is then weighted by the cost of transitioning from state l to state k. It is also interesting to note that the Viterbi algorithm and forward algorithm largely share the same recursion. The only difference between the two algorithms lies in the fact that the Viterbi algorithm uses the maximum function whereas the forward algorithm uses a sum.

We can now compute $f_k(t)$ based on a weighted sum of all the forward algorithm results tabulated during the previous time step. As shown in Figure 8.2, the forward algorithm can be easily implemented in a KxN dynamic programming table. The first column of the table is initialized according to the initial state probabilities a_{i0} and the algorithm then proceeds to process each column from left to right. Because there are KN entries and each entry examines a total of K other entries, this leads to $O(K^2N)$ time complexity and $O(KN)$ space.

In order now to calculate the total probability of a sequence of observed characters under the current HMM, we need to express this probability in terms of the forward algorithm gives in the following way:

$$P(x_1, \ldots, x_n) = \sum_l P(x_1, \ldots, x_n, \pi_N = l) = \sum_l f_l(N) \tag{8.5}$$

Hence, the sum of the elements in the last column of the dynamic programming table provides the total probability of an observed sequence of characters. In practice, given a sufficiently long sequence of emitted characters, the forward probabilities decrease very rapidly. To circumvent issues associated with storing small floating point numbers, logs-probabilities are used in the calculations instead of the probabilities themselves. This alteration requires a slight adjustment to the algorithm and the use of a Taylor series expansion for the exponential function.

8.1.5 This lecture

- This lecture will discuss **posterior decoding**, an algorithm which again will infer the hidden state sequence π that maximizes a different metric. In particular, it finds the most likely state at every position over all possible paths and does so using both the **forward** and **backward** algorithm.
- Afterwards, we will show how to encode "memory" in a Markov chain by adding more states to search a genome for dinucleotide CpG islands.
- We will then discuss how to use Maximum Likelihood parameter estimation for supervised learning with a labelled dataset
- We will also briefly see how to use Viterbi learning for unsupervised estimation of the parameters of an unlabelled dataset
- Finally, we will learn how to use Expectation Maximization (EM) for unsupervised estimation of parameters of an unlabelled dataset where the specific algorithm for HMMs is known as the Baum-Welch algorithm.

8.2 Posterior Decoding

8.2.1 Motivation

Although the **Viterbi decoding** algorithm provides one means of estimating the hidden states underlying a sequence of observed characters, another valid means of inference is provided by **posterior decoding**.

Posterior decoding provides the most likely state at any point in time. To gain some intuition for posterior decoding, let's see how it applies to the situation in which a dishonest casino alternates between a fair and loaded die. Suppose we enter the casino knowing that the unfair die is used 60 percent of the time. With this knowledge and no die rolls, our best guess for the current die is obviously the loaded one. After one roll, the probability that the loaded die was used is given by

$$P(die = loaded|roll = k) = \frac{P(die = loaded) * P(roll = k|die = loaded)}{P(roll = k)}. \tag{8.6}$$

If we instead observed a sequence of N die rolls, how do perform a similar sort of inference? By allowing information to flow between the N rolls and influence the probability of each state, posterior decoding is a natural extension of the above inference to a sequence of arbitrary length. More formally, instead of identifying a single path of maximum likelihood, posterior decoding considers the probability of any path lying in state k at time t given all of the observed characters, i.e. $P(\pi_t = k|x_1, \ldots, x_n)$. The state that maximizes this probability for a given time is then considered as the most likely state at that point.

It is important to note that in addition to information flowing forward to determine the most likely state at a point, information may also flow backward from the end of the sequence to that state to augment or reduce the likelihood of each state at that point. This is partly a natural consequence of the reversibility of Bayes' rule. To elucidate this, imagine the casino example again. As stated earlier, without observing any rolls, the $state_0$ is most likely to be unfair. If the first roll is a 6, our belief that $state_1$ is unfair is reinforced (if rolling sixes is more likely in an unfair die). If a 6 is rolled again, information flow backwards from the second die roll and reinforces our $state_1$ belief of an unfair die even more. The more rolls we have, the more information that flows backwards and reinforces or contrasts our beliefs about the state thus illustrating the way information flows backward and forward to affect our belief about the states in Posterior Decoding.

Using some elementary manipulations, we can rearrange this probability into the following form using Bayes' rule:

$$\pi_t^* = argmax_{\pi_t} P(\pi_t = k|x_1, \ldots, x_n) = argmax_{\pi_t} \frac{P(\pi_t = k, x_1, \ldots, x_n)}{P(x_1, \ldots, x_n)} \tag{8.7}$$

Because $P(x)$ is a constant, we can neglect it when maximizing the function. Therefore,

$$\pi_t^* = argmax_{\pi_t} P(\pi_t = k, x_1, \ldots, x_t) * P(x_{t+1}, \ldots, x_n|\pi_t = k, x_1, \ldots, x_t) \tag{8.8}$$

Using the Markov property, we can simply write this expression as follows:

$$\pi_t^* = argmax_{\pi_t} P(\pi_t = k, x_1, \ldots, x_t) * P(x_{t+1}, \ldots, x_n|\pi_t = k) = argmax_{\pi_t} f_k(t) * b_k(t) \tag{8.9}$$

Here, we've defined $f_k(t) = P(\pi_t = k, x_1, \ldots, x_t)$ and $b_k(t) = P(x_{t+1}, \ldots, x_n|\pi_t = k)$. As we will shortly see, these parameters are calculated using the **forward algorithm** and the **backward algorithm** respectively. To solve the posterior decoding problem, we merely need to solve each of these subproblems. The forward algorithm has been illustrated in the previous chapter and in the review at the start of this chapter and the backward algorithm will be explained in the next section.

Before explaining the backward algorithm, one may ask if Posterior Decoding can provide a path like the Viterbi algorithm would. The answer is yes; since Posterior Decoding can provide the most likely state for every point, assembling these states provides the path. Later on, we would compare the pros and cons of the paths generated using Viterbi algorithm and Posterior Decoding.

8.2.2 Backward Algorithm

As previously described, the backward algorithm is used to calculate the following probability:

$$b_k(t) = P(x_{t+1}, \ldots, x_n|\pi_t = k) \tag{8.10}$$

We can begin to develop a recursion n by expanding into the following form:

$$b_k(t) = \sum_l P(x_{t+1}, \ldots, x_n, \pi_{t+1} = l | \pi_t = k) \tag{8.11}$$

From the Markov property, we then obtain:

$$b_k(t) = \sum_l P(x_{t+2}, \ldots, x_n | \pi_{t+1} = l) * P(\pi_{t+1} = l | \pi_t = k) * P(x_{t+1} | \pi_{t+1} = k) \tag{8.12}$$

The first term merely corresponds to $b_l(t+1)$. Expressing in terms of emission and transition probabilities gives the final recursion:

$$b_k(t) = \sum_l b_l(i+1) * a_{kl} * e_l(x_{t+1}) \tag{8.13}$$

Comparison of the forward and backward recursions leads to some interesting insight. Whereas the forward algorithm uses the results at $t-1$ to calculate the result for t, the backward algorithm uses the results from $t+1$, leading naturally to their respective names. Another significant difference lies in the emission probabilities; while the emissions for the forward algorithm occur from the current state and can therefore be excluded from the summation, the emissions for the backward algorithm occur at time $t+1$ and therefore must be included within the summation.

Given their similarities, it is not surprising that the backward algorithm is also implemented using a KxN dynamic programming table. The algorithm, as depicted in Figure 8.3, begins by initializing the rightmost column of the table to unity. Proceeding from right to left, each column is then calculated by taking a weighted sum of the values in the column to the right according to the recursion outlined above. After calculating the leftmost column, all of the backward probabilities have been calculated and the algorithm terminates. Because there are KN entries and each entry examines a total of K other entries, this leads to $O(K^2N)$ time complexity and $O(KN)$ space, bounds identical to those of the forward algorithm.

Just as P(X) was calculated by summing the rightmost column of the forward algorithm's DP table, P(X) can also be calculated from the sum of the leftmost column of the backward algorithm's DP table. Therefore, these methods are virtually interchangeable for this particular calculation.

Figure 8.3: The Backward Algorithm

Did You Know?

Note that even when executing the backward algorithm, forward transition probabilities are used i.e if moving in the backward direction involves a transition from state B $\rightarrow$ A, the probability of transitioning from state A $\rightarrow$ B is used

8.2.3 The Big Picture

Given that we can calculate both $f_k(t)$ and $b_k(t)$ in $\theta(K^2N)$ time and $\theta(KN)$ space for all $t = 1 \dots n$, we can use posterior decoding to determine the most likely state π_t^* for $t = 1 \dots n$. The relevant expression is given by

$$\pi_t^* = argmax_k P(\pi_i = k|x) = \frac{f_k(i) * b_k(i)}{P(x)} \tag{8.14}$$

With two methods (Viterbi and posterior) to decode, which is more appropriate? When trying to classify each hidden state, the Posterior decoding method is more informative because it takes into account all possible paths when determining the most likely state. In contrast, the Viterbi method only takes into account one path, which may end up representing a minimal fraction of the total probability. At the same time, however, posterior decoding may give an invalid sequence of states! For example, the states identified at time points t and $t + 1$ might have zero transition probability between them. As a result, selecting a decoding method is highly dependent on the application of interest.

FAQ

Q: What does it imply when the Viterbi algorithm and Posterior decoding disagree on the path?

A: In a sense, it is simply a reminder that our model of the world can never be correct. In the genomic context, it might be a result of some 'funky' biology; alternative splicing, for instance. In some cases, the Viterbi algorithm will be close to the Posterior decoding while in some others they may disagree.

8.3 Encoding Memory in a HMM: Detection of CpG islands

CpG islands are defined as regions within a genome that are enriched with pairs of C and G nucleotides on the same strand. Typically, when this dinucleotide is present within a genome, it becomes methylated and cytosine mutates into a thymine, thereby rendering CpG dinucleotides relatively rare. Because the methylation can occur on either strand, CpG's usually mutate into a TpG or a CpA. However, when situated within an active promoter, methylation is suppressed and CpG dinucleotides are able to persist. Similarly, CpGs in regions important to cell function are conserved due to evolutionary pressure. As a result, detecting CpG islands can highlight promoter regions and other important regions within a genome.

Did You Know?

CpG stands for [C]ytosine - [p]hosphate backbone - [G]uanine. The 'p' implies that we are referring to the same strand of the double helix

Given their biological significance, CpG islands are prime candidates for modelling. Initially, one may attempt to identify these islands by scanning the genome for fixed intervals rich in GC. This approach's efficacy is undermined by the selection of an appropriate window size; while too small of a window may not capture all of a particular CpG island, too large of a window would result in the dilution of the CpG island's effects. Examining the genome on a per codon basis also leads to difficulties because CpG pairs do not necessarily lie within one reading frame. Instead, HMMs are much better suited to modelling this scenario because, as we shall shortly see in unsupervised learning, HMMs can adapt their underlying parameters to maximize their likelihood.

Not all HMMs, however, are well suited to this particular task. An HMM model that only considers the single nucleotide frequencies of C's and G's will fail to capture the nature of CpG islands. Consider one such HMM with the two following hidden states :

- '+' state representing CpG islands
- '-' state: representing non-islands

Each of these two states then emits A, C, G and T bases with a certain probability. Although the CpG islands in this model can be enriched with C's and G's by increasing their respective emission probabilities, this model will fail to capture the fact that the C's and G's predominantly occur in pairs.

Because of the Markov property that governs HMM's, the only information available at each time step must be contained within the current state. Therefore, to encode memory within a Markov chain, we need to augment the state space. To do so, the individual '+' and '-' states can be replaced with 4 '+' states and 4 '-' states: A+, C+, G+, T+, A-, C-, G-, T- (Figure 8.4). Specifically, there are 2 ways to model this, and this choice will result in different emission probabilities:

- One model suggests that the state A+, for instance, implies that we are currently in a CpG island and the *previous* character was an A. The emission probabilities here will carry most of the information and the transitions will be fairly degenerate.
- Another model suggests that the state A+, for instance, implies that we are currently in a CpG island and the *current* character is an A. The emission probability here will be 1 for A and 0 for all other letters and the transition probabilities will bear most of the information in the model and the emissions will be fairly degenerate. We will assume this model from now on.

Did You Know?

The number of transitions is the square of the number of states. This gives a rough idea of how increasing HMM "memory" (and hence states) scale.

The memory of this system derives from the fact that each state can only emit one character and therefore "remembers" its emitted character. Furthermore, the dinucleotide nature of the CpG islands is incorporated within the transition matrices. In particular, the transition frequency from C_+ to G_+ states is significantly higher than from C_- to a G_- states, demonstrating that these pairs occur more often within the islands.

FAQ

Q: Since each state emits only one character, can we then say this reduces to a Markov Chain instead of a HMM?

A: No. Even though the emissions indicate the letter of the hidden state, they do not indicate if the state is a CpG island or not.

FAQ

Q: How do we incorporate our knowledge about the system while training HMM models eg. some emission probabilities of 0 in the CpG island detection case?

A: We could either force our knowledge on the model by setting some parameters and leaving others to vary or we could let the HMM loose on the model and let it discover those relationships. As a matter of fact, there are even methods that simplify the model by forcing a subset of parameters to be 0 but allowing the HMM to choose which subset.

Given the above framework, we can use posterior decoding to analyze each base within a genome and determine whether it is most likely a constituent of a CpG island or not. But having constructed the expanded HMM model, how can we verify that it is in fact better than the single nucleotide model? We previously demonstrated that the forward or backward algorithm can be used to calculate $P(x)$ for a given

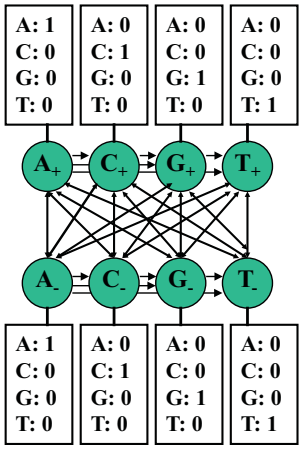


Figure 8.4: HMM for CpG Islands

model. If the likelihood of our dataset is higher given the second model than the first model, it most likely captures the underlying behavior more effectively.

FAQ

Q: Are there other ways to encode the memory for CpG island detection?

A: Other ideas that may be experimented with include

- Emit dinucleotides and figure out a way to deal with overlap.
- Add a special state that goes from C to G.

8.4 Learning

We saw how to score and decode an HMM-generated sequence in two different ways. However, these methods assumed that we already knew the emission and transition probabilities. Fortunately, the HMM framework enables the learning of these probabilities when provided a set of training data.

When the training data is labelled, estimation of the probabilities is a form of supervised learning. One such instance would occur if we were given a DNA sequence of one million nucleotides in which the CpG islands had all been experimentally annotated and were asked to use this to estimate our model parameters

In contrast, when the training data is unlabelled, the estimation problem is a form of unsupervised learning. Continuing with the CpG island example, this situation would occur if the provided DNA sequence contained no island annotation whatsoever and we needed to both estimate model parameters and identify the islands.

8.4.1 Supervised Learning

When provided with labelled data, estimating model parameters is actually trivial. Suppose that you are given a labelled sequence $x_1, \ldots, x_N$ as well as the true hidden state sequence $\pi_1, \ldots, \pi_N$. Intuitively, one might expect that the probabilities that maximize the data's likelihood are the actual probabilities that one observes within the data. This is indeed the case and can be formalized by defining A_{kl} to be the number of times hidden state k transitions to l and $E_k(b)$ to be the number of times b is emitted from hidden state k.

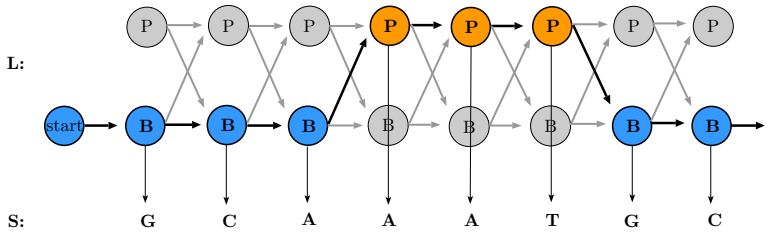


Figure 8.5: Supervised Learning of CpG islands

The parameters θ that maximize $P(x|\theta)$ are simply obtained by counting as follows:

$$a_{kl} = \frac{A_{kl}}{\sum_i A_{ki}} \tag{8.15a}$$

$$e_k(b) = \frac{E_k(b)}{\sum_c E_k(c)} \tag{8.15b}$$

One example training set is shown in Figure 8.5. In this example, it is obvious that the probability of transitioning from B to P is $\frac{1}{3+1} = \frac{1}{4}$ (there are 3 B to B transitions and 1 B to P transitions) and the probability of emitting a G from the B state is $\frac{2}{2+2+1} = \frac{2}{5}$ (there are 2 G's emitted from the B state, 2 C's and 1 A)

Notice, however, that in the above example the emission probability of character T from state B is 0 because no such emissions were encountered in the training set. A zero probability, either for transitioning or emitting, is particularly problematic because it leads to an infinite log penalty. In reality, however, the zero probability may merely have arisen due to over-fitting or a small sample size. To rectify this issue and maintain flexibility within our model, we use pseudocounts instead of absolute counts.

- $A^*_{kl} = A_{kl} + r_{kl}$
- $E_k(b)^* = E_k(b) + r_k(b)$

Larger pseudocount parameters correspond to a strong prior belief whereas small pseudocount parameters ($r << 1$) are used to avoid 0 probabilities.

8.4.2 Unsupervised Learning

Unsupervised learning involves estimating parameters based on unlabelled data. An iterative approach is typically most suitable to solving these types of problems. Suppose we have some sort of prior belief about what each emission and transition probability should be. Given these parameters, we can use a decoding method to infer the hidden states underlying the provided data sequence. Using this particular decoding parse, we can then re-estimate the transition and emission counts and probabilities in a process similar to that used for supervised learning. If we repeat this procedure until the improvement in the data's likelihood remains relatively stable, the data sequence should ultimately drive the parameters to their appropriate values.

FAQ

Q: Why does unsupervised learning even work? Or is it magic?

A: Unsupervised learning works because we have the sequence (input data) and this guides every step of the iteration; to go from a labelled sequence to a set of parameters, the later are guided by the input and its annotation, while to annotate the input data, the parameters and the sequence guide the procedure.

For HMM's in particular, two main methods of unsupervised learning are useful.

Expectation Maximization using Viterbi training

The first method, **Viterbi training**, is relatively simple but not entirely rigorous. After picking some initial best-guess model parameters, it proceeds as follows:

E step: Perform Viterbi decoding to find $\pi^\star$
Calculate A^*_{kl}, $E_k(b)^*$ using pseudocounts based on the transitions and emissions observed in $\pi^\star$ states given the latest parameters and observed sequence (Expectation step)

M step: Calculate the new parameters a_{kl}, $e_k(b)$ using the simple counting formalism in supervised learning (Maximization step)

Iteration: Repeat the E and M steps until the likelihood $P(x|\theta)$ converges

Although Viterbi training converges rapidly, its resulting parameter estimations are usually inferior to those of the Baum-Welch Algorithm. This result stems from the fact that Viterbi training only considers the most probable hidden path instead of the collection of all possible hidden paths.

Expectation Maximization: The Baum-Welch Algorithm

The more rigorous approach to unsupervised learning involves an application of Expectation Maximization to HMM's. In general, EM proceeds in the following manner:

Init: Initialize the parameters to some best-guess state

E step: Estimate the expected probability of hidden states given the latest parameters and observed sequence (Expectation step)

M step: Choose new maximum likelihood parameters using the probability distribution of hidden states (Maximization step)

Iteration: Repeat the E and M steps until the likelihood of the data given the parameters converges

The power of EM lies in the fact that $P(x|\theta)$ is guaranteed to increase with each iteration of the algorithm. Therefore, when this probability converges, a local maximum has been reached. As a result, if we utilize a variety of initialization states, we will most likely be able to identify the global maximum, i.e. the best parameters θ. The Baum-Welch algorithm generalizes EM to HMM's. In particular, it uses the forward and backward algorithms to calculate $P(x|\theta)$ and to estimate A_{kl} and $E_k(b)$. The algorithm proceeds as follows:

Initialization
1. Initialize the parameters to some best-guess state

Iteration
1. Run the forward algorithm
2. Run the backward algorithm
3. Calculate the new log-likelihood $P(x|\theta)$
4. Calculate A_{kl} and $E_k(b)$
5. Calculate a_{kl} and $e_k(b)$ using the pseudocount formulas
6. Repeat until $P(x|\theta)$ converges

Previously, we discussed how to compute $P(x|\theta)$ using either the forward or backward algorithm's final results. But how do we estimate A_{kl} and $E_k(b)$? Let's consider the expected number of transitions from state k to state l given a current set of parameters θ. We can express this expectation as

$$A_{kl} = \sum_t P(\pi_t = k, \pi_{t+1} = l|x, \theta) = \sum_t \frac{P(\pi_t = k, \pi_{t+1} = l, x|\theta)}{P(x|\theta)} \tag{8.16}$$

Exploiting the Markov property and the definitions of the emission and transition probabilities leads to the following derivation:

$$A_{kl} = \sum_t \frac{P(x_1 \dots x_t, \pi_t = k, \pi_{t+1} = l, x_{t+1} \dots x_N | \theta)}{P(x|\theta)} \tag{8.17a}$$

$$= \sum_t \frac{P(x_1 \dots x_t, \pi_t = k) * P(\pi_{t+1} = l, x_{t+1} \dots x_N | \pi_t, \theta)}{P(x|\theta)} \tag{8.17b}$$

$$= \sum_t \frac{f_k(t) * P(\pi_{t+1} = l | \pi_t = k) * P(x_{t+1} | \pi_{t+1} = l) * P(x_{t+2} \dots x_N | \pi_{t+1} = l, \theta)}{P(x|\theta)} \tag{8.17c}$$

$$\Rightarrow A_{kl} = \sum_t \frac{f_k(t) * a_{kl} * e_l(x_{t+1}) * b_l(t+1)}{P(x|\theta)} \tag{8.17d}$$

A similar derivation leads to the following expression for $E_k(b)$:

$$E_k(b) = \sum_{i|x_i=b} \frac{f_k(t) * b_k(t)}{P(x|\theta)} \tag{8.18}$$

Therefore, by running the forward and backward algorithms, we have all of the information necessary to calculate $P(x|\theta)$ and to update the emission and transition probabilities during each iteration. Because these updates are constant time operations once $P(x|\theta)$, $f_k(t)$ and $b_k(t)$ have been computed, the total time complexity for this version of unsupervised learning is $\theta(K^2NS)$, where S is the total number of iterations.

FAQ

Q: How do you encode your prior beliefs when learning with Baum-Welch?

A: Those prior beliefs are encoded in the initializations of the forward and backward algorithms

8.5 Current Research Directions

- HMM's have been used extensively in various fields of computational biology. One of the first such applications was in a gene-finding algorithm known as GENSCAN written by Chris Burge and Samuel Karlin [1]. Because the geometric length distribution of HMM's does not model exonic regions well, Burge et. al used an adaptation of HMM's known as hidden semi-Markov models (HSMM's). These types of models differ in that whenever a hidden state is reached, the length of duration of that state (d_i) is chosen from a distribution and the state then emits exactly d_i characters. The transition from this hidden state to the next is then analogous to the HMM procedure except that $a_{kk} = 0$ for all k, thereby preventing self-transitioning. Many of the same algorithms that were previously developed for HMM's can be modified for HSMM's. Although the details won't be discussed here, the forward and backward algorithms can be modified to run in $O(K^2N^3)$ time, where N is the number of observed characters. This time complexity assumes that there is no upper bound on the length of a state's duration, but imposing such a bound reduces the complexity to $O(K^2ND^2)$, where D is the maximum possible duration of a state.

  The basic state diagram underlying Burge's model is depicted in Figure 8.6. The included diagram only lists the states on the forward strand of DNA, but in reality a mirror image of these states is also included for the reverse strand, resulting in a total of 27 hidden states. As the diagram illustrates, the model incorporates many of the major functional units of genes, including exons, introns, promoters, UTR's and poly-A tails. In addition, three different intronic and exonic states are used to ensure that the total length of all exons in a gene is a multiple of three. Similar to the CpG island example, this expanded state-space enabled the encoding of memory within the model.

- An interesting subject that may be explored also concerns the agreement of Viterbi and Posterior decoding paths; not just for CpG island detection but even for chromatin state detection. One may look at multiple paths by sampling, asking questions such as:
  - What is the maximum a posteriori vs viterbi path? Where do they differ?
  - Can complete but maximally disjoint (from Viterbi) paths be found?

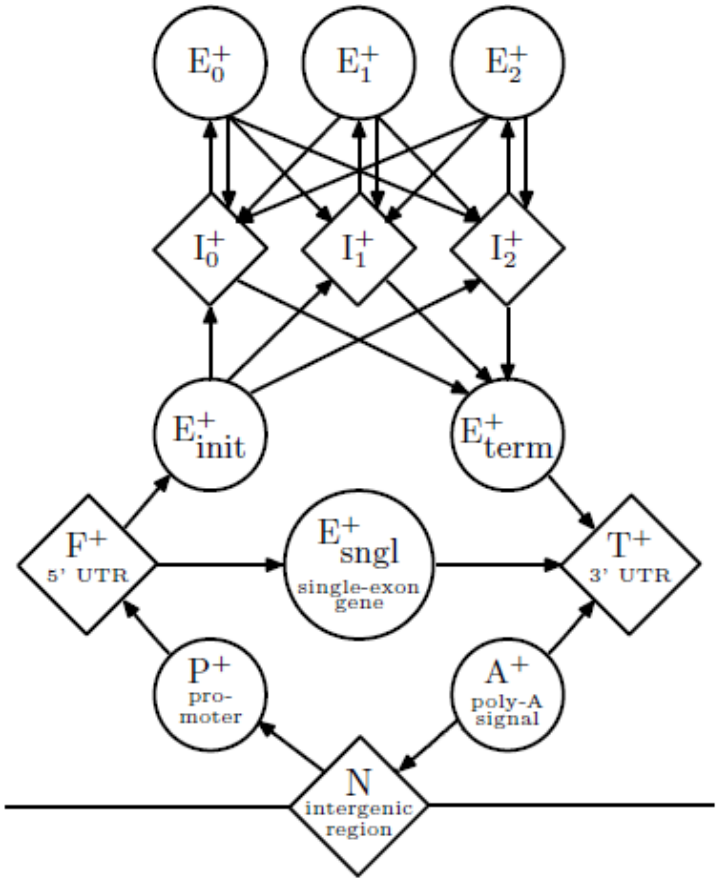


Figure 8.6: State Space Diagram used in GENSCAN

8.6 Further Reading

8.7 Tools and Techniques

8.8 What Have We Learned?

Using the basic computational framework provided by Hidden Markov Models, we've learned how to infer the most likely set of hidden states underlying a sequence of observed characters. In particular, a combination of the forward and backward algorithms enabled one form of this inference, i.e. posterior decoding, in $O(KN^2)$ time. We also learned how either unsupervised or supervised learning can be used to identify the best parameters for an HMM when provided with an unlabelled or labelled dataset. The combination of these decoding and parameter estimation methods enable the application of HMM's to a wide variety of problems in computational biology, of which CpG island and gene identification form a small subset. Given the flexibility and analytical power provided by HMM's, these methods will play an important role in computational biology for the foreseeable future.

Bibliography

[1] Christopher B Burge and Samuel Karlin. Finding the genes in genomic dna. *Current Opinion in Structural Biology*, 8(3):346 – 354, 1998.

CHAPTER
NINE

GENE IDENTIFICATION: GENE STRUCTURE, SEMI-MARKOV, CRFS

Tim Helbig (2011)
Jenny Cheng (2010)

Figures

9.1 Introduction

After a genome has been sequenced, a common next step is to attempt to infer the functional potential of the organism or cell encoded through careful analysis of that sequence. This mainly takes the form of identifying the protein coding genes within the sequence as they are thought to be the primary units of function within living systems; this is not to say that they are the only functional units within genomes as things such as regulatory motifs and non-coding RNAs are also imperative elements.

This annotation of the protein coding regions is too laborious to do by hand, so it is automated in a process known as computational gene identification. The algorithms underlying this process are often based on Hidden Markov Models (HMMs), a concept discussed in previous chapters to solve simple problems such as knowing whether a casino is rolling a fair versus a loaded die. Genomes, however, are very complicated sets of data, replete with long repeats, overlapping genes (where one or more nucleotides are part of two or more distinct genes) and pseudogenes (non-transcribed regions that look very similar to genes) among many other obfuscations. Thus, experimental and evolutionary data often needs to be included into HMMs for greater annotational accuracy, which can result in a loss of scalability or a reliance on incorrect assumptions of independence. Alternative algorithms have been utilized to address the problems of HMMs including those based on Conditional Random Fields (CRFs), which rely on creating a distribution of the hidden states of the genomic sequence in question conditioned on known data. Use of CRFs has not phased out HMMs as both are used with varying degrees of success in practice.[1]

[1] R. Guigo (1997). "Computational gene identification: an open problem." Computers Chem. Vol. 21.

9.2 Overview of Chapter Contents

This chapter will begin with a discussion of the complexities of the Eukaryotic gene. It will then describe how HMMs can be used as a model to parse Eukaryotic genomes into protein coding genes and regions that are not; this will include reference to the strengths and weaknesses of an HMM approach. Finally, the use of CRFs to annotate protein coding regions will be described as an alternative.

9.3 Eukaryotic Genes: An Introduction

Within eukaryotic genomes, only a small fraction of the nucleotide content actually consists of protein coding genes (in humans, protein coding regions make up about 1%-1.5% of the entire genome). The rest of the DNA is classified as intergenic regions (See Figure 9.1) and contains things such as regulatory motifs, transposons, integrons and non-protein coding genes.[2]

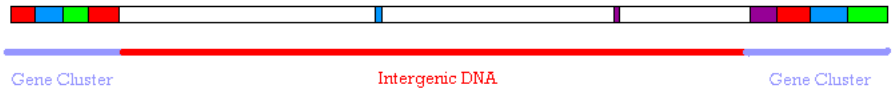


Figure 9.1: Intergenic DNA

Further, of the small fraction of the DNA that is transcribed into mRNA, not all of it is translated into protein. Certain regions known as introns, are removed or "spliced" out of the precursor mRNA. This now processed mRNA, containing only "exons" and some other additional modifications discussed in previous chapters, is translated into protein. (See Figure 9.2) The goal of computational gene identification is thus not only to pick out the few regions of the entire Eukaryotic genome that encode for proteins but also to parse those protein coding regions into identities of exon or intron so that the sequence of the synthesized protein can be known.

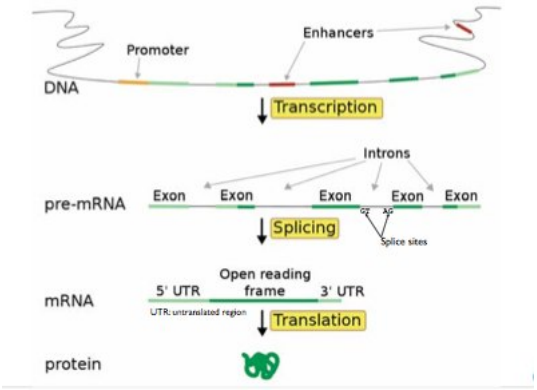


Figure 9.2: Intron/Exon Splicing

9.4 Assumptions for Computational Gene Identification

The general assumptions for computational gene identification are that exons are delineated by a sequence AG at the start of the exon and a sequence of GT at the end of the exon. For protein-coding genes, the start codon (ATG) and the end codons (TAA, TGA, TAG) delineate the open reading frame. (Most of these ideas can be seen in Figure 9.3) These assumptions will be incorporated into more complex HMMs described below.

[2] "Intergenic region." http://en.wikipedia.org/wiki/Intergenic_region

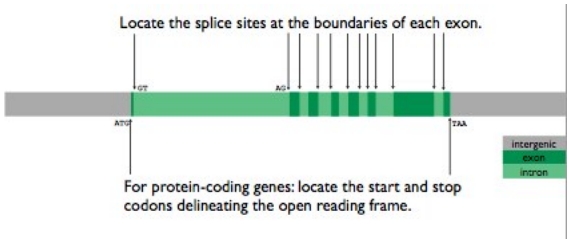


Figure 9.3: Delineation of Exons and Open Reading Frames

9.5 Hidden Markov Models

A toy Hidden Markov Model is a generative approach to model this behavior. Each emission of the HMM is one DNA base/letter. The hidden states of the model are intergenic, exon, intron. Improving upon this model would involve including hidden states DonorG and DonorT. The DonorG and DonorT states utilize the information that exons are delineated by GT at the end of the sequence before the start of an intron. (See Figure 9.4 for inclusion of DonorG and DonorT into the model)

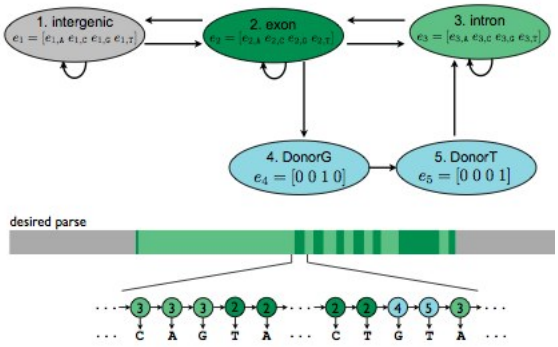


Figure 9.4: Hidden Markov Model Utilizing GT Donor Assumption

The e in each state represents emission probabilities and the arrows indicate the transition probabilities.

Aside from the initial assumptions, additional evidence such as evolutionary conservation and experimental mRNA data can help create an HMM to better model the behavior. (See Figure 9.5)

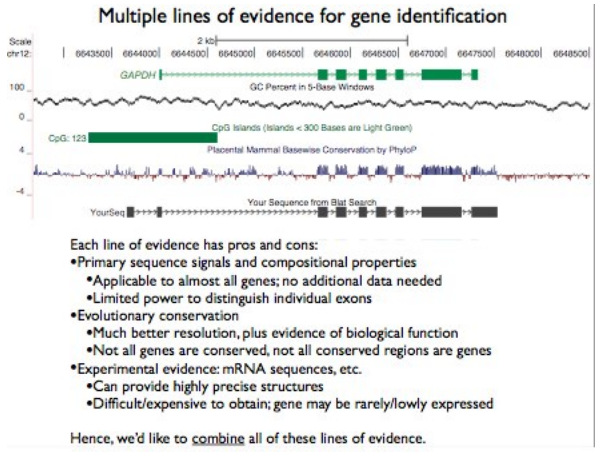


Figure 9.5: Multiple lines of evidence for gene identification

Combining all the lines of evidence discussed above, we can create an HMM with composite emissions in that each emitted value is a "tuple" of collected values. (See Figure 9.6)

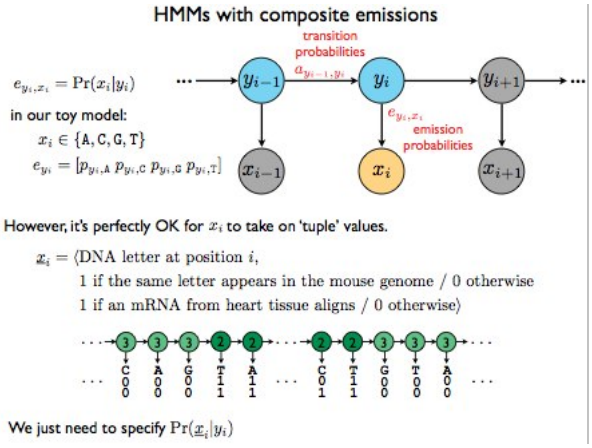


Figure 9.6: HMMs with composite emissions

A few assumptions of this composite model are that each new emission "feature" is independent of the rest. However, this creates the problem that with each new feature, the tuple increases in length, and the number of states of the HMM increases exponentially, leading to a combinatorial explosion, which thus means poor scaling. (Examples of more complex HMMs that can result in poor scaling can be found in Figure 9.7)

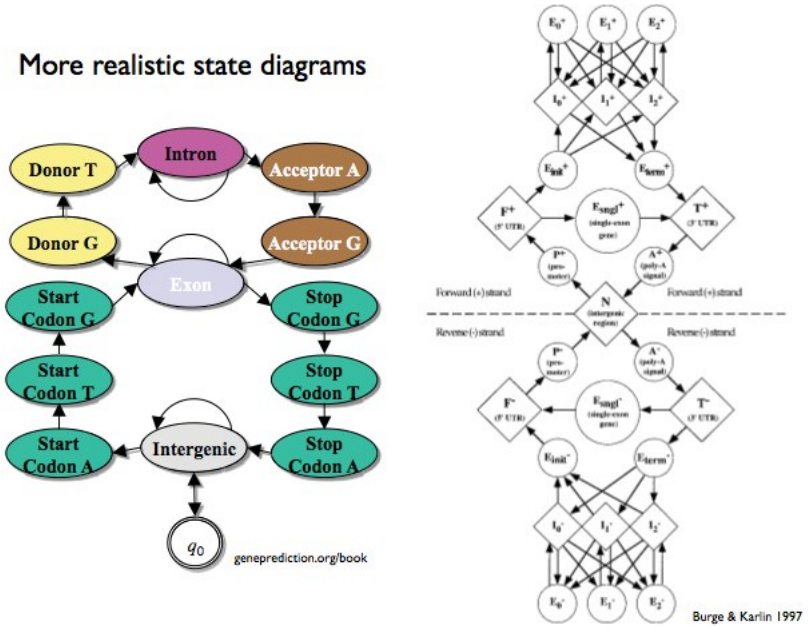


Figure 9.7: State diagram that considers direction of RNA translation

9.6 Conditional Random Fields

Conditional Random Fields, CRFs, are an alternative to HMMs. Being a discriminative approach, this type of model doesnt take into account the joint distribution of everything, as does a poorly scaling HMM. The hidden states in a CRF are conditioned on the input sequence. (See Figure 9.8)[3]

[3]Conditional Random Field. Wikipedia. http://en.wikipedia.org/wiki/Conditional_random_field

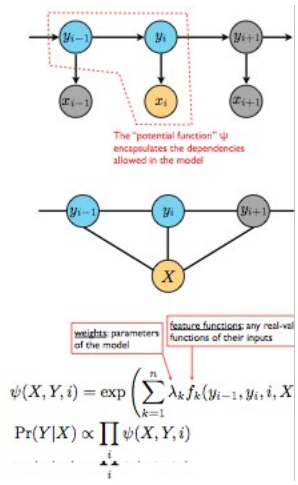


Figure 9.8: Conditional random fields: a discriminative approach conditioned on the input sequence

A feature function is like a score, returning a real-valued number as a function of its inputs that reflects the evidence for a label at a particular position. (See Figure 9.9) The conditional probability of the emitted sequence is its score divided by the total score of the hidden state. (See Figure 9.10)

$$f_1(y_{i-1}, y_i, i, X) = \begin{cases} 1 & \text{if } y_i = \texttt{exon} \text{ and position } i \text{ is conserved in mouse} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(y_{i-1}, y_i, i, X) = \begin{cases} 1 & \text{if } y_i = \texttt{exon} \text{ and position } i \text{ is conserved in rat} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(y_{i-1}, y_i, i, X) = \#\text{ of mRNA sequences aligning to position } i \text{ (if } y_i = \texttt{exon}\text{; 0 otherwise)}$$

Figure 9.9: Examples of feature functions

$$\Pr(Y|X) = \frac{1}{Z(X)} \prod_i \psi(X, Y, i) \quad \text{where} \quad Z(X) = \sum_{Y'} \prod_i \psi(X, Y', i)$$

Figure 9.10: Conditional probability score of an emitted sequence

Each feature function is weighted, so that during the training, the weights can be set accordingly.

The feature functions can incorporate vast amounts of evidence without the Naive Bayes assumption of independence, making them both scalable and accurate. However, training is much more difficult with CRFs than HMMs.

9.7 Other Methods

Besides HMMs and CRFs, other methods do exist for computational gene identification. Semi-markov models generate variable sequence length emissions, meaning that the transitions are not entirely memory-less on the hidden states.

Max-min models are adaptations of support vector machines. These methods have not yet been applied to mammalian genomes. [4]

[4]For better understanding of SVM: http://dspace.mit.edu/bitstream/handle/1721.1/39663/6-034Fall-2002/OcwWeb/Electrical-Engineering-and-Computer-Science/6-034Artificial-IntelligenceFall2002/Tools/detail/svmachine.htm

9.8 Conclusion

Computational gene identification, because it entails finding the functional elements encoded within a genome, has a lot of practical significance as well as theoretical significance for the advancement of biological fields.

The two approaches described above are summarized below in Figure 9.11:

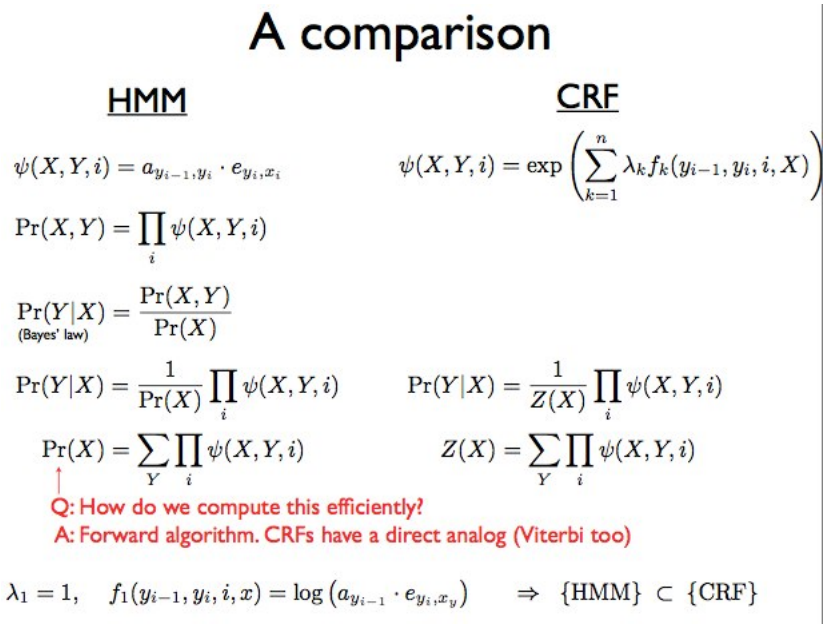


Figure 9.11: A comparison of HMMs and CRFs

9.8.1 HMM

- generative model
- randomly generates observable data, usually with a hidden state
- specifies a joint probability distribution
- $P(x, y) = P(x|y)P(y)$
- sometimes hard to model dependencies correctly
- hidden states are the labels for each DNA base/letter
- composite emissions are a combination of the DNA base/letter being emitted with additional evidence

9.8.2 CRF

- discriminative model
- models dependence of unobserved variable y on an observed variable x
- $P(y|x)$
- hard to train without supervision
- more effective for when the model doesnt require joint distribution

In practice, the resulting gene specification using CONTRAST, a CRF implementation, is about 46.2% at its maximum. This is because in biology, there are a lot of exceptions to the standard model, such as overlapping genes, nested genes, and alternative splicing. Having models include all of those exceptions sometimes yields worse predictions; this is a non-trivial tradeoff. However, technology is improving and within the next five years, there will be more experimental data to fuel the development of computational gene identification, which in turn will help generate a better understanding of the syntax of DNA.

9.9 Current Research Directions

9.10 Further Reading

9.11 Tools and Techniques

9.12 What Have We Learned?

Bibliography

1.R. Guigo (1997). "Computational gene identification: an open problem."

2."Intergenic region." http://en.wikipedia.org/wiki/Intergenic_region

3.Conditional Random Field. Wikipedia. http://en.wikipedia.org/wiki/Conditional_random_field

4.http://dspace.mit.edu/bitstream/handle/1721.1/39663/6-034Fall-2002/OcwWeb/Electrical-Engineering-and-Computer-Science/6-034Artificial-IntelligenceFall2002/Tools/detail/svmachine.htm

CHAPTER

TEN

RNA FOLDING

Guest Lecture by
Stefan Washietl (wash@mit.edu)
Scribed by Sam Sinaei (samsinai@mit.edu) (2010
) Scribed by Archit Bhise (archit@mit.edu) (2012)

Figures

10.1 Motivation and Purpose

RNA **(Ribonucleic acid)** is a very important molecule. The aim for this chapter is to understand the methods that can explain, or even predict the secondary structure of RNA.

To accomplish this, we first look at RNA from a biological perspective and explain the known biological roles of RNA. Then, we study the different methods that exist to predict RNA structure. There are two main approaches to the RNA folding problem: 1) predicting the RNA structure based on thermodynamic stability of the molecule, and looking for a thermodynamic optimum 2) probabilistic models which try to find the states of the RNA molecule in a probabilistic optimum.

Finally, we can use evolutionary data in order to increase the confidence of our predictions by these methods.

10.2 Chemistry of RNA

RNA is consisted of a 5-carbon sugar, ribose, which is attached to an organic base (either adenine, uracil, cytosine or guanine). The biochemical difference between DNA and RNA is in two places : 1) the 5-carbon sugar has no hydroxyl group in the 5 position 2) the uracil presence in the RNA which is the non-methylated form of thymine instead of just thymine. The presence of ribose in RNA makes its structure more flexible than DNA, therefore the RNA molecule is able to fold and make bonds within itself which makes the single stranded RNA more stable than single stranded DNA.

10.3 Origin and Functions of RNA

The initial belief about the RNA was that it acts as an intermediate between the DNA code and the protein, which is true. However, in early 80s, the discovery of catalytic RNAs (ribozymes) expanded the perspective on what this molecule can actually do in living things. Sidney Altman and Thomas Cech discovered the first ribozyme, RNase P which is able to cleave off the head of tRNA. Self-splicing introns (group I introns) were also one of the first ribozymes that were discovered. They do not need any protein as catalysts to splice. Single or double stranded RNA also serves as the information storage and replication agent in some viruses.

The **RNA World Hypothesis** proposed by Gilbert which tries to explain the origin of life, relies on the fact that RNA can have both information storage, and catalytic activity at the same time, which are the fundamental characteristics that a living system needs to have. In short, it says in the beginning RNA molecules were the first replicators, and because they were catalytic at the same time, it was possible for them to replicate without dependency on other molecules. Although to this day, there are no natural self-replicating RNA found in vivo, self-replicating RNA molecules have been created in lab via artificial selection. For example, a chimeric construct of a natural ligase ribozyme with an in vitro selected template binding domain has been shown to be able to replicate at least one turn of an RNA helix.

Through evolution, RNA has passed its information storage role to DNA, because it is more stable and less prone to mutation and acted as an intermediate between DNA and proteins, which took over some of RNAs catalytic role in the cell. Thus, scientists sometimes refer to RNA as molecular fossils. Nevertheless, RNA still plays an important catalytic role in the living organisms. For instance, the catalytic portion of the ribosome i.e. the main functional part of the ribosomal complex consists of RNA. RNA also has regulatory roles in the cell, and basically serves as an agent for the cell to sense and react to the environment.

10.3.1 Riboswitches

Regulatory RNAs have different families, and one of the most important ones are **riboswitches**. Riboswitches are involved in different levels of gene regulation. In some bacteria, important regulations are done by simple RNA families. One example is the thermosensor in Listeria, a riboswitch that blocks the ribosomes at low temperature (since the hydrogen bonds are more stable). The RNA then forms a semi-double stranded conformation which does not bind to the ribosome and turns the ribosome off. At higher temperatures (37 C), the double strand opens up and allows ribosome to attach to a certain region in the riboswitch, making translation possible once again. Another famous Riboswitch is the adenine Riboswitch (and in general purine riboswitches) , which regulate protein synthesis. For example the ydhl mRNA which has a terminator stem at the end and blocks it from translation, but when the Adenine concentration increases in the cell, it binds to the mRNA and changes its conformation such that the terminator stem disappears.

10.3.2 microRNAs

There are other sorts of RNAs such as **microRNAs**, a more modern variant of RNA (relatively). Their discovery unveiled a novel non-protein layer of gene regulation (e.g. the EVF-2 and HOTAIR miRNAs). EVF-2 is interesting because its transcribed from an ultra conserved enhancer, and separates from the transcription string by forming a hairpin, and thereafter returns to the very same enhancer (along with a protein Dlx-2) and regulates its activity. HOTAIR RNA induces changes in chromatin state, and regulates the methylation of Histones, which in turn silences the HOX-D cluster.

10.3.3 Other types of RNA

We can also look at types of **noncoding RNAs**.

piRNAs are the largest class of small non-coding RNA molecules in animals. They are primarily involved in the silencing of transposons, but likely have a lot of functions. They are also involved in epigenetic modications, and post-transcriptional gene silencing.

lncRNAs are long transcripts produced that operate functionally as RNAs and are not translated into proteins. Many studies implicate lncRNAs in epigenetic modications, maybe acting as a targeting mechanism or as a molecular scaffold for Polycomb proteins. lncRNAs are likely to possess numerous functions, many are nuclear, many are cytoplasmic.

10.4 RNA Structure

We have learned about different functions of RNA, and it should be clear by now how fundamental the role of RNA in living systems is. Because it is impossible to understand how RNA actually does all these activities in the cell, without knowing what its structure is, in this part we will look into the structure of RNA.

RNA structure can be studied in three different levels 10.1:

1. *Primary* structure: the sequence in which the bases (U, A, C, G) are aligned.
2. *Seconary* structure: the 2-D analysis of the [hydrogen] bonds between different parts of RNA. In other words, where RNA becomes double-stranded, where RNA forms a hairpin or a loop or other similar forms.
3. *Tertiary* structure: the complete 3-D structure of RNA, i.e. how the string bends, where it twists and such.

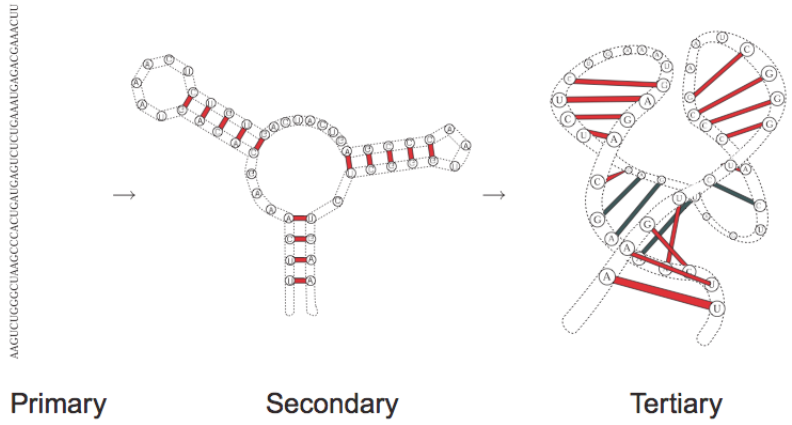


Figure 10.1: Graphical representation of the hierarchy of RNA strucure complexity

As mentioned before, the presence of ribose in RNA enables it to fold and create double-helixes with itself. The primary structure is fairly easy to obtain through sequencing the RNA. We are mainly interested in understanding the secondary structure for RNA: where the loops and hydrogen bonds form and create the functional attributes of RNA. Ideally, we would like to study the tertiary structure because this is the final state of the RNA, and what gives it its true functionality. However, the tertiary structure is very hard to compute and beyond the scope of this lecture.

Even though studying the secondary structure can be tricky, there are some simple ideas that work quite well in predicting it. Unlike proteins, in RNA, most of the stabilizing]free energy for the molecule comes from

its secondary structure (rather than tertiary in case of proteins). RNAs initially fold into their secondary structure and then form their tertiary structure, and therefore there are very interesting facts that we can learn about a certain RNA molecule by just knowing its secondary structure.

Finally, another great property of the secondary structure is that it is usually well conserved in evolution, which helps us improve the secondary structure predictions and also to find ncRNA (non-coding RNA)s. There are widely used representations for the secondary structure of RNA:

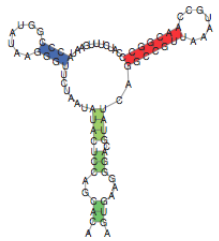


Figure 10.2: The typical representation of RNA secondary structure in textbooks. It clearly shows the secondary substructure in RNA.

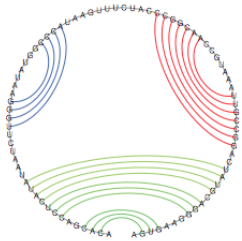


Figure 10.3: Graph drawing where the back-bone is a circle and the base pairings are the arcs within the circle. Note that the graph is outer-planar, meaning the arcs do not cross.

Formally: A secondary structure is a vertex labeled graph on n vertices with an adjacency matrix $A = (a_{ij})$ fulfilling:

- $a_{i,i+1} = 1 for 1 \leq i \leq n1$ (continuous backbone)
- For each $i, 1 \leq i \leq N$ there is at most one $a_{ij} = 1$ where $j \gtrless i + / - 1$(a base only forms a pair with one other at the time)
- If $a_{ij} = a_{kl} = 1 and i < k < j then i < l < j$ (ignore pseudo knots)

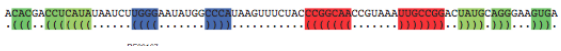


Figure 10.4: A machine readable dot-bracket notation, in which for each paired nucleotide you open a bracket(and close it when you reach its match) and for each unpaired element you have a dot.

10.5 RNA Folding Problem and Approaches

Finally, we get to the point where we want to study the RNA structure. The goal here is to predict the secondary structure of the RNA, given its primary structure (or its sequence). The good news is we can

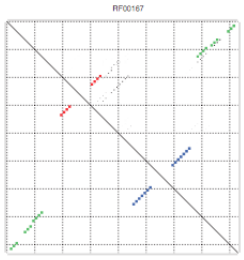


Figure 10.5: A matrix representation, in which you have a dot for each pair.

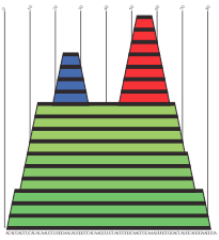


Figure 10.6: Mountain plot, in which for pairs you go one step up in the plot and if not you go one step to the right.

find the optimal structure using dynamic programming. Now in order to set up our dynamic programming framework we would need a scoring scheme, which we would create using the contribution of each base pairing to the physical stability of the molecule. In other words, we want to create a structure with minimum free energy, in in our simple model we would assign each base pair an energy value. 10.7

	A	C	G	U
A	+10	+10	+10	−2
C	+10	+10	−3	+10
G	+10	−3	+10	−1
U	−2	+10	−1	+10

Figure 10.7: Example of a scoring scheme for base pair matches. Note that G-U can form a wobble pair in RNA.

The optimum structure is going to be the one with a minimum free energy and by convention negative energy is stabilizing, and positive energy is non-stabilizing. Using this framework, we can use dynamic programming (DP) to calculate the optimal structure because 1) this scoring scheme is additive 2) we disallowed pseudo knots, which means we can divide the RNA into two smaller ones which are independent, and solve the problem for these smaller RNAs.

We want to find a DP matrix E_{ij}, in which we calculate the minimum free energy for subsequence i to j. The first approach to this is Nussinov's algorithm.

10.5.1 Nussinov's algorithm

The recursion formula for this problem was first described by Nussinov in 1978. 10.8

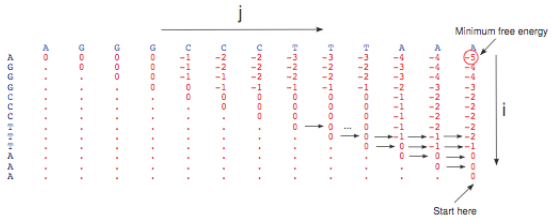
It calculates the best substructure for the subsequences and then builds them up to larger sequences till it finds the structure of the whole sequence. There are basically only two cases to get from one step to the

$$E_{ij} = \min\left\{E_{i+1,j}, \min_{k,\Pi_{ik}=1}\left\{E_{i+1,k-1} + E_{k+1,j} + \beta_{ik}\right\}\right\}$$

- E_{ij} . . . Minimum energy of subsequence $i \ldots j$
- β_{ij} . . . Energy contribution of pair (i,j)
- Π_{ij} is 1 if bases i and j can pair and 0 otherwise.

Figure 10.8: The recursion formula for Nussinov algorithm, along with a graphical depiction of how it works.

next in the recursion: 1) The newly added base is unpaired 2) It is paired with some base k. For the latter case the base pair (i, k) divides the problem into two subproblems which can be then recursively solved the same way. Below 10.9 is example for the DP Matrix after running this algorithm:



- Folding the sequence AGGGCCCTTTAAA
- Simple energy model with $\beta_{ij} = -1$ for all base-pairs (i.e. finds maximum matching structure)
- Backtracking finds the following optimal structure: (.((.)))((.))

Figure 10.9: The algorithm starts by assigning the diagonal for the matrix as 0 (since you cannot pair with yourself) and then works through the recursion up and right, the minimum free energy is the top-rightmost element in the matrix. The minimum length for a loop is 1 here (usually 3). As i counts backward form n to 1, j counts from 1 to n. In this example we simply assign -1 for a pair, and 0 for a non-pair.

When you calculate the minimum free energy, you are interested in the sequence which corresponds to this particular energy, a helper matrix is filled to backtracking over the sequence,

Here is the code for the backtracking:

```
function Backtrack(i,j)
begin
    if i > j then return
    if K_ij = 0 then Backtrack(i + 1,j) else
        output: (i, K_ij)
        Backtrack(i + 1,K_ij − 1)
        Backtrack(K_ij + 1,j)
    end
end
```

Figure 10.10: The helper array K_{ij} is filled during the recursion that holds the optimal secondary structure when k is paired with i for a sub-sequence $i..j$. If i is unpaired in the optimal structure, K_{ij} is filled during the recursion that holds the optimal secondary structure when k is paired with i for a sub-sequence $i...j$. If i is unpaired in the optimal structure, $K_{ij} = 0$.

This model is very simplistic and there are some limitations to it. Most importantly, stacking interaction between neighboring pairs is a very important factor (even more important than the hydrogen bonds)in RNA folding which is not considered by the Nussinovs model. 10.11

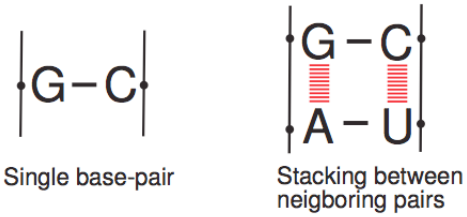


Figure 10.11: Stacking between neighboring base pairs is RNA. The flat aromatic structure of the base causes quantum interactions between stacked bases and changes its physical stability.

Therefore people have thought of methods to integrate such biophysical factors into our prediction. One improvement for instance is that instead of assigning energies to single base pairs, we assign them to faces of the graph (structural elements in 10.12). In order to find out the total energy of the structure, we have to find the free energy of each substructure, and simply add them up. The stacking energies can be calculated by melting oligonucleotides experimentally.

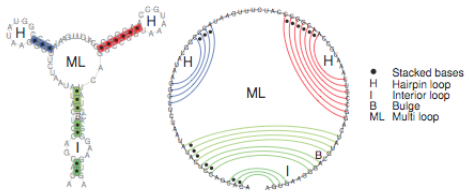


Figure 10.12: Various internal substructures in a folded RNA. A hairpin is consisted of a terminal loop connected to a paired region, an internal loop is an unpaired region within the paired region. A Bulge is a special case of an interior loop with a single mis-pair. a Multi loop is a loop which consists of multiple of these components (in this example two hairpins and a paired region, all connected to a loop).

10.5.2 Zuker Algorithm

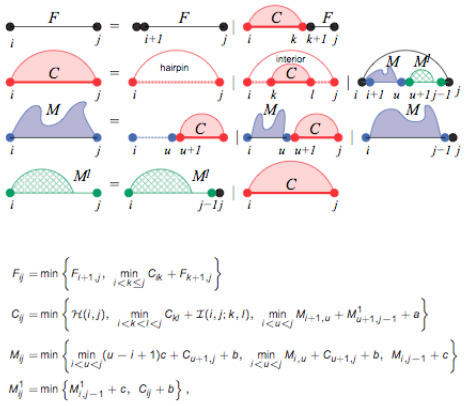
Theefore, we use a variant which includes stacking energies to calculate the RNA structure. This is called the Zuker algorithm. Like Nussinovs, it assumes that the optimal structure is the one with the lowest equilibrium free energy. Nevertheless, it includes the total energy contributions from the various substructures which is partially determined by the stacking energy. Some modern RNA folding algorithms use this algorithm for RNA structure predictions.

In the Zuker algorithm, we have four different cases to deal with. Figure 10.13 shows a graphical outline of the decomposition steps. The procedure requires four matrices. F_{ij} contains the free energy of the overall optimal structure of the subsequence x_{ij}. The newly added base can be unpaired or it can form a pair. For the latter case, we introduce the helper matrix C_{ij}, that contains the free energy of the optimal substructure of x_{ij} under the constraint that i and j are paired. This structure closed by a base-pair can either be a hairpin, an interior loop or a multi-loop.

The hairpin case is trivial because no further decomposition is necessary. The interior loop case is also simple because it reduces again to the same decomposition step. The multi-loop step is more complicated.

The energy of a multi loop depends on the number of components, i.e. substructures that emanate from the loop. To implicitly keep track of this number, there is a need for two additional helper matrices. M_{ij} holds the free energy of the optimal structure of x_{ij} under the constraint that x_{ij} is part of a multi loop with at least one component. M^1_{ij} holds the free energy of the optimal structure of x_{ij} under the constraint that x_{ij} is part of a multi-loop and has exactly one component closed by pair (i, k) with $i < k < j$. The idea is to decompose a multi loop in two arbitrary parts of which the first is a multi-loop with at least one component and the second a multi-loop with exactly one component and starting with a base-pair.

These two parts corresponding to M and M^1 can further be decomposed into substructures that we already know, i.e. unpaired intervals, substructures closed by a base-pair,or (shorter) multi-loops. (The recursions are also summarized in 10.13.



$$F_{ij} = \min\left\{F_{i+1,j},\ \min_{i<k\le j} C_{ik} + F_{k+1,j}\right\}$$

$$C_{ij} = \min\left\{\mathcal{H}(i,j),\ \min_{i<k<l<j} C_{kl} + \mathcal{I}(i,j;k,l),\ \min_{i<u<j} M_{i+1,u} + M^1_{u+1,j-1} + a\right\}$$

$$M_{ij} = \min\left\{\min_{i<u<j}(u-i+1)c + C_{u+1,j} + b,\ \min_{i<u<j} M_{i,u} + C_{u+1,j} + b,\ M_{i,j-1} + c\right\}$$

$$M^1_{ij} = \min\left\{M^1_{i,j-1} + c,\ C_{ij} + b\right\},$$

Figure 10.13: F describes the unpaired case, C is described by one of the three conditions : hairpin,interior loop, or a composition of structures i.e. a multi loop. M^1 is a multi loop with only one component, where are M might have multiple of them. The | icon is notation for *"or"*.

In reality, however, at room temperature (or cell temperature), RNA is not actually in one single state, but rather varies in a Thermodynamic ensemble of structure. Base pairs can break their bonds quite easily, and although we might find an absolute optimum in terms of free energy, it might be the case that there is another sub-optimal structure which is very different from what e predicted and has an important role in the cell. To fix the problem we can calculate the base pair probabilities to get the ensemble of structures, and then we can have a much better idea of what the RNA structure probably looks like. In order to do this, we utilize the Boltzman factor:

$$\text{Prob}(\mathcal{S}) = \frac{\exp(-\Delta G(\mathcal{S})/RT)}{Z}$$

This gives us the probability of a given structure, in a thermodynamic system. We need to normalize the temperature using the partition function Z,which is the weighted sum of all structures, based on their Boltzman factor:

$$Z = \sum_{\mathcal{S}} \exp(-\Delta G(\mathcal{S})/RT)$$

We can also represent this ensemble graphically, using a dot plot to visualize the base pair probabilities. To calculate the specific probability for a base pair (i, j) , we need to calculate the partition function, which is given by the following formula :

To calculate Z (the partition function over the whole structure), we use the recursion similar to the

$$p_{ij} = \frac{\hat{Z}_{ij} Z_{i+1,j-1} \exp(-\beta_{ij}/RT)}{Z}$$

Nussinovs Algorithm (known as McCaskill Algorithm).The inner partition function is calculated using the formula:

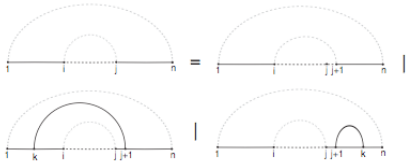
$$Z_{ij} = Z_{i+1,j} + \sum_{\substack{i+1 \le k \le j \\ \Pi_{ik}=1}} Z_{i+1,k-1} Z_{k+1,j} \exp(-\beta_{ik}/RT)$$

With each of the additions corresponding to a different split in our sequence as the next figure illustrates. Note that the addition are multiplied to the energy functions since it is expressed as a exponential.

Similarly the outer partition function is calculated with a the same idea using the formula:

$$\hat{Z}_{ij} = \hat{Z}_{i,j+1} + \sum_{\substack{1 \le k < i \\ \Pi_{k,j+1}=1}} \hat{Z}_{k,j+1} \exp(-\beta_{k,j+1}/RT) Z_{k+1,i-1} + \sum_{\substack{j+2 \le k \le n \\ \Pi_{k,j+1}=1}} \hat{Z}_{i,k} \exp(-\beta_{k,j+1}/RT) Z_{j+2,k-1}$$

corresponding to different splits in the area outside the base pairs (i, j).



10.6 Evolution of RNA

It is useful to understand the evolution of RNA structure, because it unveils valuable data, and can also give us hints to refine our structure predictions. When we look into functionally important RNAs over time, we realize their nucleotides have changed at some parts, but their structure is well-conserved.

In RNA there are a lot of **compensatory mutations and consistent mutations**. In a consistent mutation, the structure doesnt change e.g. an AU pair mutates to form a G pair. In a compensatory mutation there are actually two mutations, one disrupts the structure, but the second mutation restores it, for example an AU pair changes to a CU which does not pair well, but in turn the U mutates to a G to restore a CG pair. In an ideal world, if we have this knowledge, this is the be the key to predict the RNA structure, because evolution never lies. We can calculate the **mutual information content** for two different RNAs and compare it. In other words, you compare the probabilities of two base pair structures agreeing randomly vs. if they have evolved to be conserve the structure.

The mutual information content is calculated via this formula:

$$MI_{ij} = \sum_{X,Y} f_{ij}(XY) \log \frac{f_{ij}(XY)}{f_i(X) f_j(Y)}$$

If we normalize these probabilities, and store the MI in bits, we can plot it in a 3D model and track the evolutionary signatures. In fact, this was the method for determining the structure of ribosomal RNAs long before they were found by crystallography.

The real problem is that we dont have so much information, so what we usually do is combine the folding prediction methods with phylogenetic information in order to get a reliable prediction. The most common way to do this is to combine to Zuker algorithm with some covariance scores. For example, we add stabilizing energy if we have a compensatory mutation, and destabilizing energy if we have a single nucleotide mutation.

10.7 Probabilistic Approach to the RNA Folding Problem

RNA-coding sequence inside the genome Finding RNA-coding sequences inside the genome is a very hard problem. However there are ways to do it. One way is to combine the thermodynamic stability information, with a normalized RNAfold score and then we can do a Support Vector Machine (SVM) classification, and compare the thermodynamic stability of the sequence to some random sequences of the same GC content and the same length and see how many standard deviations is the given structure more stable that the expected value.
We can combine it with the evolutionary measure and see if the RNA is more conserved or not. This gives us (with relative accuracy) an idea if the genomic sequence is actually coding an RNA.

We have studied only half of the story. Although the thermodynamic approach is a good way (and the classic way) of folding the RNAs, some part of the community like to study it from a different aspect.

Lets assume for now that we dont know anything about the physics of RNA or the Boltzman factor. Instead, we look into the RNA as a string of letters for which we want to find the most probable structure. We have already learned about the Hidden Markov Models in the previous lectures. They are a nice way to make predictions about the hidden states of a probabilistic system. The question is can we use Hidden Markov models for the RNA folding problem? The answer is yes.

We can represent RNA structure as a set of hidden states of dots and brackets (recall the dot-bracket representation of RNA in part 3). There is an important observation to make here: the positions and the pairings inside the RNA are not independent, so we cannot simply have a state of an opening bracket without any considerations of the events that are happening downstream.

Therefore we need to extend the HMM framework to allow for nested correlations. Fortunately, the probabilistic framework to deal with such a problem already exists. It is known as stochastic context-free grammar (SCFG).

Context Free Grammar in a nutshell
You have:

- Finite set of non-terminal symbols (states) e.g. $\{A, B, C\}$ and terminal symbols e.g. $\{a, b, c\}$
- Finite set of Production rules. e.g. $\{A \to aB, B \to AC, B \to aa, \to ab\}$
- An initial (start) nonterminal

You want to find a way to get from one state to another (or to a terminal). $A \to aB \to aAC \to aaaC \to aaaab$
In a stochastic CFG, the only difference is that each relation has a certain probability.e.g.$P(B \to AC) = 0.25$ $P(B \to aa) = 0.75$

Phylogenetic evaluation is easily combined with SCFGs, since there are many probabilistic models for phylogenetic data. The Probabilistic models are not discussed in detail in this lecture but the following picture basically gives an analogy between the Stochastic models and the methods that we have see so far in the class.

- Analogies to thermodynamic folding:
  - CYK $\leftrightarrow$ Minimum Free energy (Nussinov/Zuker)
  - Inside/outside algorithm $\leftrightarrow$ Partition functions (McCaskill)
- Analogies to Hidden Markov models:
  - CYK Minimum $\leftrightarrow$ Viterbi's algorithm
  - Inside/outside algorithm $\leftrightarrow$ Forward/backwards algorithm
- Given a parameterized SCFG (Θ, Ω) and a sequence x, the Cocke-Younger-Kasami (CYK) dynamic programming algorithm finds an optimal (maximum probability) parse tree $\hat{\pi}$:

  $\hat{\pi} = argmax Prob(\pi, x|\Theta, \Omega)$
- The *Inside algorithm*, is used to obtain the total probability of the sequence given the model summed ovver all parse trees,

  $Prob(x|\Theta, \Omega) = \Sigma Prob(x, \pi|\Theta, \Omega)$

10.7.1 Application of SCFGs

- Consensus secondary structure prediction: Pfold
  - First Phylo-SCFG
- Structural RNA gene nding: EvoFold
  - Uses Pfold grammar
  - Two competing models:
    - Non-structural model with all columns treated as evolving independently
    - Structural model with dependent and independent columns
  - Sophisticated parametrization

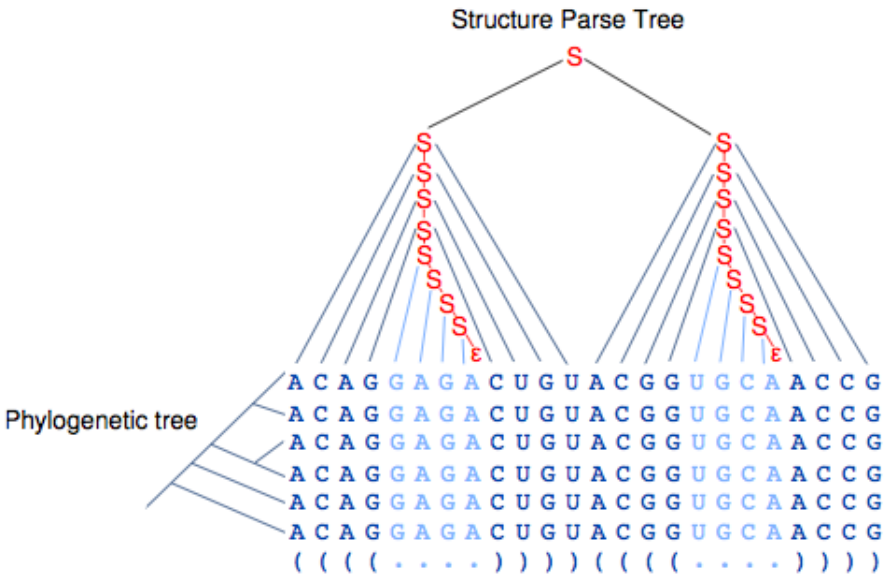


Figure 10.14: A) Single sequence: Terminal symbols are bases or base-pairs, Emission probabilities are base frequencies in loops and paired regions B) Phylo-SCFG: Terminal symbols are single or paired alignment columns, Emission probabilities calculated from phylogenetic model and tree using Felsenstein's algorithmWe to try to better understand RNA-RNA interactions.

10.8 Advanced topics

There still remain a host of other problems that need to be solved by studying RNA structure. This section will profile some of them.

10.8.1 Other problems

Observe some of the problems depicted graphically below:

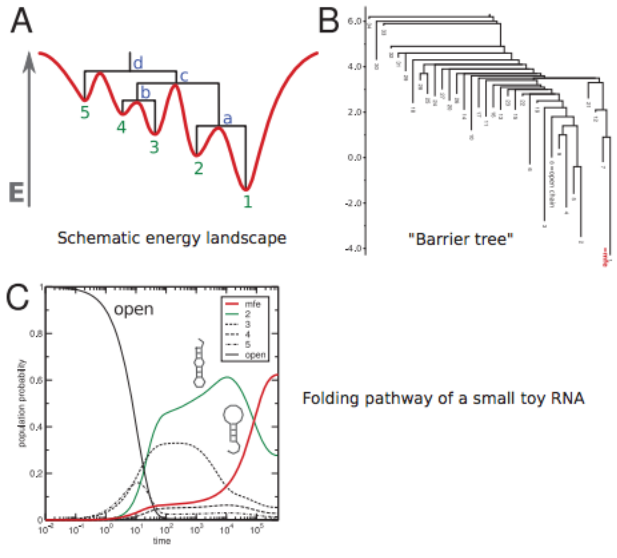


Figure 10.15: We can study kinetics and folding pathways in further depth.

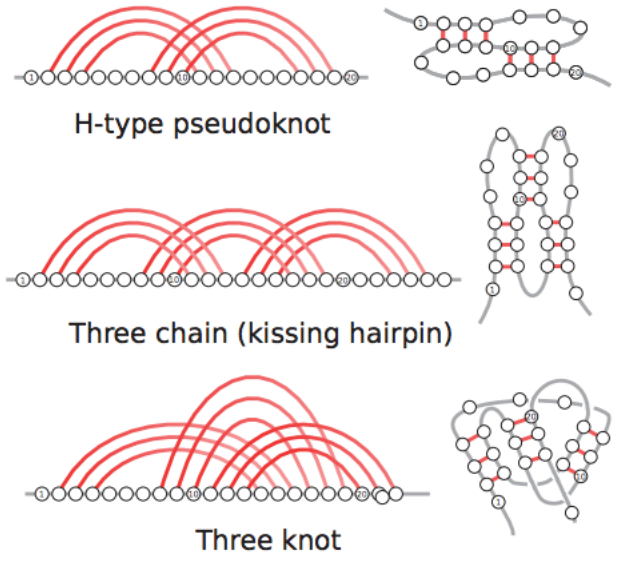


Figure 10.16: We can investigate pseudoknots.

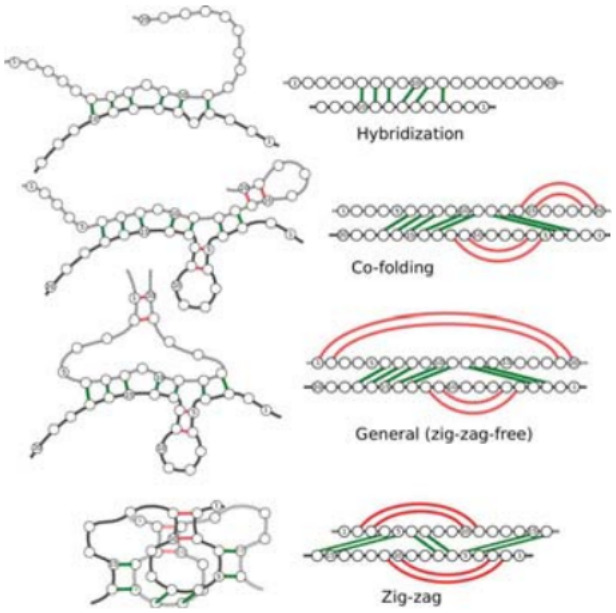


Figure 10.17: We can try to better understand RNA-RNA interactions.

10.8.2 Relevance

There are plenty of RNAs inside the cell aside from mRNAs, tRNAs and rRNAs. The question is what is the relevance of all this non-coding RNA? Some believe it is noise resulted through experiment, some think its just biological noise that doesnt have a meaning in the living organism. On the other hand some believe junk RNA might actually have an important role as signals inside the cell and all of it is actually functional, the truth probably lies somewhere in between.

10.8.3 Current research

There are conserved regions in the genome that do not code any proteins, and now Stefans et al. are looking into them to see if they have structures that are stable enough to form functional RNAs. It turns out that around 6% of these regions have hallmarks of good RNA structure, which is still 30000 structural elements. The group has annotated some of these elements, but there is still a long way to go. a lot of miRNA, snowRNAs have been found and of course lots of false positives. But there exciting results coming up in this topic! so the final note is, it's a very good area to work in!

10.9 Summary and key points

1. The functional spectrum of RNAs is practically unlimited
   (a) RNAs similar to contemporary Ribozymes and Riboswitches might have existed in an RNA world. Some of them still exist as living fossils in current cells.
   (b) Evolutionarily younger RNAs including miRNAs and many long ncRNAs form a non-protein based regulatory layer.

2. RNA structure is critical for their function and can be predicted computationally
   (a) Nussinov/Zuker: Minimum Free Energy structure
   (b) McCaskill: Partition function and pair probabilities
   (c) CYK/Inside-Outside: probabilistic solution to the problem using SCFGs
3. Phylogenetic information can improve structure prediction
4. Computational biology of RNAs is an active eld of research with many hard algorithmic problems still open

10.10 Further reading

- Overview
  - Washietl S, Will S. et al. Computational analysis of noncoding RNAs. Wiley Interdiscip Rev RNA. 2012, 10.1002/wrna.1134
- RNA function: review papers by John Mattick
- Single sequence RNA folding
  - Nussinov R, Jacobson AB, Fast algorithm for predicting the secondary structure of single-stranded RNA.Proc Natl Acad Sci U S A. 1980 Nov; 77:(11)6309-13
  - Zuker M, Stiegler P Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic Acids Res. 1981 Jan; 9:(1)133-48
  - McCaskill JS The equilibrium partition function and base pair binding probabilities for RNA secondary structure. Biopolymers. 1990; 29:(6-7)1105-19
  - Dowell RD, Eddy SR, Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. BMC Bioinformatics. 2004 Jun; 5:71
  - Do CB, Woods DA, Batzoglou S, CONTRAfold: RNA secondary structure prediction without physics-based models. Bioinformatics. 2006 Jul; 22:(14)e90-8
- Consensus RNA folding
  - Hofacker IL, Fekete M, Stadler PF, Secondary structure prediction for aligned RNA sequences. J Mol Biol. 2002 Jun; 319:(5)1059-66
  - Knudsen B, Hein J, RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. Bioinformatics. 1999 Jun; 15:(6)446-54
- RNA gene finding
  - Pedersen JS, Bejerano G, Siepel A, Rosenbloom K, Lindblad-Toh K, Lander ES, Kent J, Miller W, Haussler D Identication and classication of conserved RNA secondary structures in the human genome. PLoS Comput Biol. 2006 Apr; 2:(4)e33
  - Washietl S, Hofacker IL, Stadler PF, Fast and reliable prediction of noncoding RNAs. Proc Natl Acad Sci U S A. 2005 Feb; 102:(7)2454-9

Bibliography

[1] R Durbin. *Biological Sequence Analysis*.

[2] W. Gilbert. "origin of life: The rna world". *Nature.*, 319(6055):618, 1986.

[3] Rachel Sealfon, 2012. Extra information taken from Recitation 5 slides.

[4] Z. Wang, M. Gestein, and M. Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nat Rev Genet.*, 10(1):57–63, 2009.

[5] Stefan Washietl, 2012. All pictures/formulas courtesy of Stefan's slides.

[6] R. Weaver. *Molecular Biology*. 3rd edition.

CHAPTER

ELEVEN

LARGE INTERGENIC NON-CODING RNAS

Guest lecture by John Rinn
Scribed by Eli Stickgold (2010)

Figures

11.1 Introduction

Epigenetics is the study of heritable changes in genetic expression and phenotype that do not result from a sequence of DNA. Each cell, despite having an identical copy of the genome, is able to differentiate into

a specialized type. There are many biological devices for accomplishing these including DNA methylation, histone modification, and various types of RNA.

DNA methylation is a binary code that is effectively equivalent to turning a gene "on" or "off". However, often times a gene might need to be more highly expressed as opposed to just being turned on. For this, histones have tails that are subject to modification. The unique combination of these two elements on a stretch of DNA can be thought of as a barcode for cell type. Even more important is the method of their preservation during replication. In the case of DNA methylation, one appropriately methylated strand is allocated to each mother or daughter cell. By leaving one trail behind, the cell is able to fill in the gaps and appropriately methylate the other cell.

As the intermediary between DNA sequences and proteins, RNA is arguably the most versatile means of regulation. As such, they will be the focus of this chapter.

Did You Know?
Cell types can be determined by histone modification or DNA methylation (a binary code, which relies on a euchromatic and heterochromatic state). These histone modifications can be thought of as a type of epigenetic barcode that allows cell DNA to be scanned for types. Non-coding RNAs called Large Intergenic Non-Coding RNAs (lincRNAs) are heavily involved in this process.

A quick history of RNA:

- **1975:** A lab testing relative levels of RNA and DNA in bull sperm discovers twice as much RNA as DNA.
- **1987:** After automated sequencing developed, weird non-coding RNAs are first found.
- **1988:** RNA is proved to be important for maintaining chromosome structures, via chromatin architecture
- **1990s:** A large number of experiments start to research
- **2000s:** Study shows Histone-methyltransferases depend on RNA, as RNAase causes the proteins to delocalize.

Transcription is a good proxy of what's active in the cell and what will turn into protein. Microarrays led to the discovery of twice as many non-coding genes as coding genes initially; now we know the ratio is even far higher than this.

11.2 Noncoding RNAs from Plants to Mammals

Basic Cycle: large RNA gets chopped up into small RNAs (siRNAs) RNA use by category:

Protists: RNA is used as a template to splice out DNA (RNA-dependent DNA elimination and splicing)

mRNA and DNA in nucleus: DNA chopped and recombined based on gaps in mRNA ("quirky phenomena")

Plants: RNA-dependent RNA polymerase, where the polymerase takes template of RNA and make a copy of it, is available in plants but not humans, and can make small RNAs. Mammals have at most one

copies. Very different than RNA polymerase and DNA polymerase in structure. From this, we know that plants do DNA methylation with noncoding RNA.

Flies: use RNAs for an RNA switch; coordinated regulation of hox gene requires noncoding RNA.

Mammals: Non-coding RNAs can form triple helices, guide proteins to them; chromatin-modifying complexes; involved in germ line; guide behaviour of transcription factors.

For the rest of this talk, we focus on specifically lincRNA, which we will define as RNA larger than 200 nucleotides.

11.2.1 Long non-coding RNAs

There are a number of different mechanisms and biological devices by which epigenetic regulation occurs. One of these is long non-coding RNAs which can be thought of as fulfilling an air traffic control function within the cell.

Long non-coding RNAs share many similar characteristics with microRNAs. They are spliced, contain multiple exons, are capped, and poly-adenuated. However, they do not have open reading frames. They look just like protein coding genes, but cannot.

They are better classified by their anatomical position:

Antisense: These are encoded on the opposite strand of a protein coding gene.

Intronic: Entirely contained with an intron of a protein coding gene.

Bidirectional: These share the same promoter as a protein coding gene, but are on the opposite side.

Intergenic: These do not overlap with any protein coding genes. Think of them as sitting blindly out in the open. They are much easier targets and will be the focus of this chapter.

11.3 Practical topic: RNAseq

RNA-seq is a method that utilizes next-generation sequencing technology to sequence cDNA allowing us to gain insight into the contents of RNA. The two main problems that RNA-seq addresses are (1) discover new genes such as splice isoforms of previously discovered genes and (2) uncover the expression levels of genes and transcripts from the sequencing data. Additionally, RNA-seq is also beginning to replace many traditional sequencing techniques allowing labs to perform experiments more efficiently.

11.3.1 How it works

The RNA-Seq machine grabs a transcript and breaks it into different fragments, where the fragments are normally distributed. With the speed that the RNA-seq can sequence these transcript fragments (or reads), there are an abundant number of reads allowing us to extract expression levels. The basic idea behind this method relies on the fact that the more abundant a transcript is, the more fragments we'll sequence from it.

The tools used to analyze RNA-Seq data are collectively known as the "Tuxedo Tools"

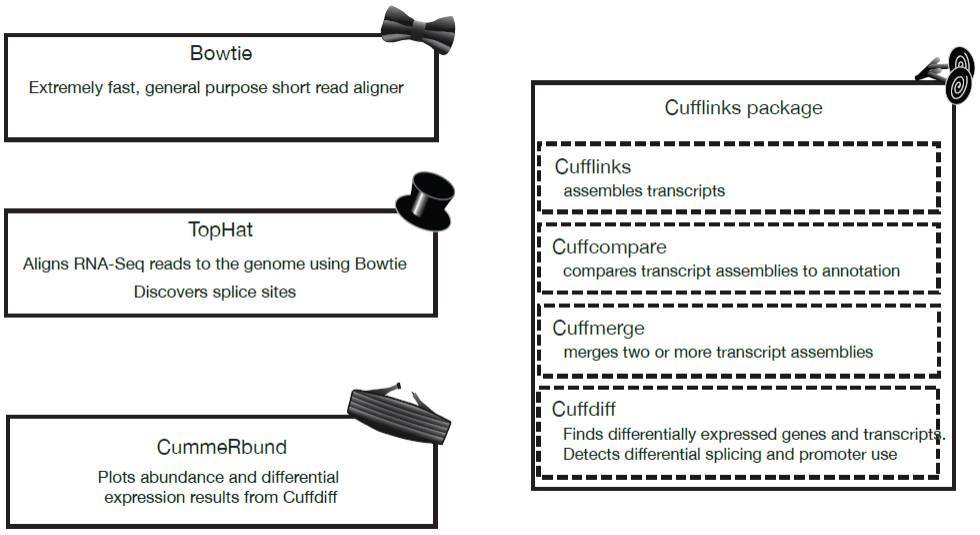


Figure 11.1: Tuxedo Tools

11.3.2 Aligning RNA-Seq reads to genomes and transcriptomes

Since RNA-Seq produces so many reads, the alignment algorithm must have a fast runtime, approximately of the order of O(n). There are two main strategies for aligning short reads, which require that we already have the transcripts.

1. Spaced seeds indexing

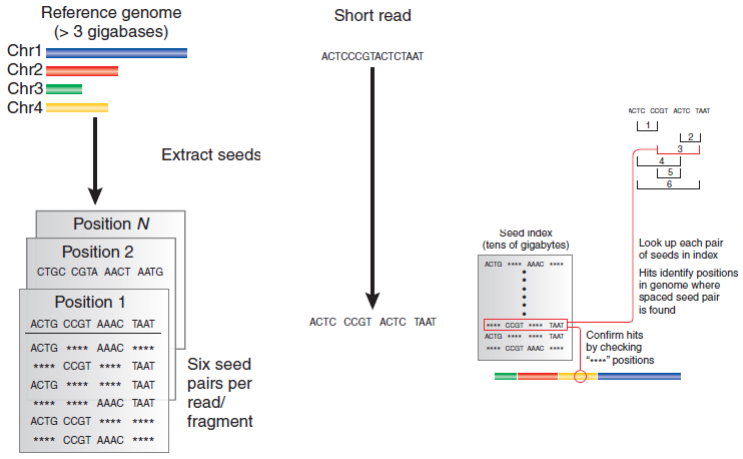


Figure 11.2: How spaced seeds indexing works

Spaced seeds indexing involves taking each read and breaking it into fragments, or "seeds". We take every combination of two fragments ("seed pairs") and compare them to an index of seeds (which will

take tens of gigabytes of space) for potential hits. Compare the other seeds to the index to make sure we have a hit.

2. Burrows-Wheeler indexing

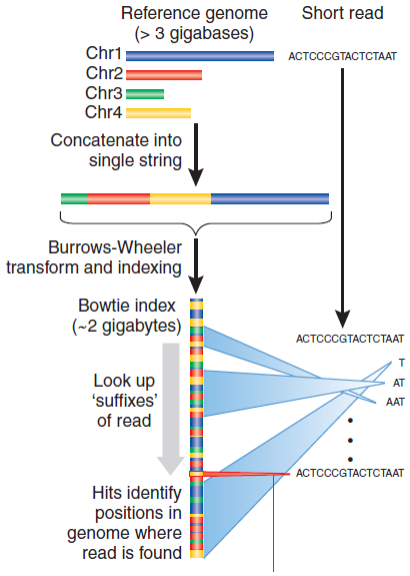


Figure 11.3: How Burrows-Wheeler indexing works

Burrows-Wheeler indexing takes the genome and scrambles it up in such a way such that you can look at the read one character at a time and throw out a huge chunk of the genome as possible alignment positions very quickly.

One major problem with these two general purpose alignment strategies is that they don't account for large gaps in alignment.

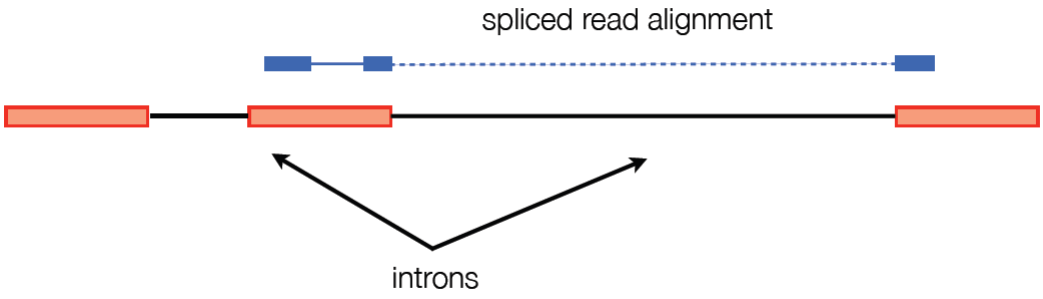


Figure 11.4: An example of a gap in alignment

To get around this, TopHat breaks the reads into smaller pieces. These pieces are aligned and reads with pieces that are mapped far apart are flagged for possible intron sites. The pieces that weren't able to be aligned are used to confirm the splice sites. The reads are then stitched back together to make full read alignments.

There are two strategies for assembling transcripts based on RNA-Seq reads.

1. Genome-guided approach (used in software such as Cufflinks)

   The idea behind this approach is that we don't necessarily know if two reads come from the same transcript, but we will know if they come from different transcripts. The algorithm is as follows: take the alignments and put them in a graph. Add an edge from $x \to y$ if x is to the left of y in the genome, x and y overlap consistently, and y is not contained in x. So we have an edge from $x \to y$ if they might come from the same transcript.

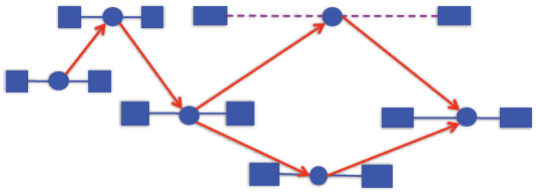


Figure 11.5: An example of how to use the graph to find transcripts

   If we walk across this graph from left to right, we get a potential transcript. Applying Dilworth's theorem to read partial orders, we can see that the size of the largest antichain in the graph is the minimum number of transcripts needed to explain the alignment. An antichain is a set of alignments with the property that no two are compatible (i.e. could arise from the same transcript)

2. Genome-independent approach (used in software such as trinity)

   The genome-independent approach attempts to piece together the transcripts directly from the reads using classical methods for overlap based read assembly, similar to the genome assembly methods.

11.3.3 Calculating expression of genes and transcripts

We want to count the number of reads from each transcript to find the expression level of the transcript. However, since we divide transcripts into equally-sized fragments, we run into the problem that longer transcripts will naturally produce more reads than a shorter transcript. To account for this, we compute expression levels in FPKM, fragments per kilobase per million fragments mapped.

Likelihood function for a gene

Suppose we sequence a particular read, call it F_1.

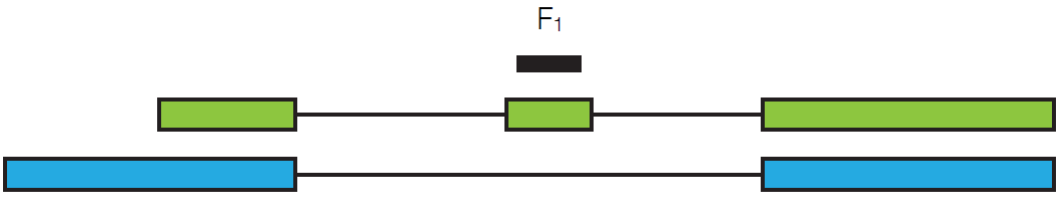


Figure 11.6: F_1

In order to get this particular read, we need to pick the particular transcript it's in and then we need to pick this particular read out from the whole transcript. If we define γ_{green} to be the relative abundance of the green transcript, then we have

$$P(F_1|\gamma_{\text{green}}) = \frac{\gamma_{\text{green}}}{l_{\text{green}}}$$

where l_{green} is the length of the green transcript. Now suppose we look at a different read, F_2.

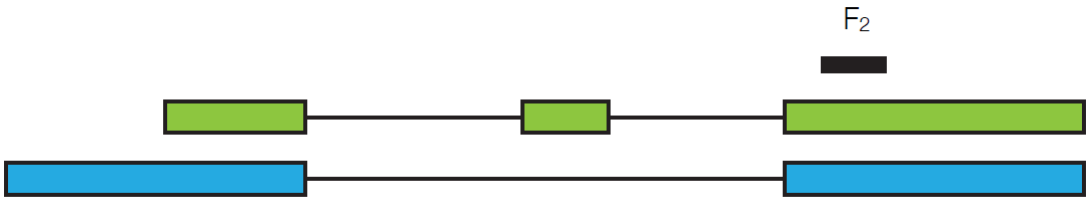


Figure 11.7: F_2

It could have come from either the green transcript of the blue transcript, so:

$$P(F_2|\gamma) = \frac{\gamma_{\text{green}}}{l_{\text{green}}} + \frac{\gamma_{\text{blue}}}{l_{\text{blue}}}$$

We can see that the probability of getting both F_1 and F_2 is just the product of the individual probabilities:

$$P(F|\gamma) = \frac{\gamma_{\text{green}}}{l_{\text{green}}} \cdot (\frac{\gamma_{\text{green}}}{l_{\text{green}}} + \frac{\gamma_{\text{blue}}}{l_{\text{blue}}})$$

We define this as our likelihood function, $L(F|\gamma)$. Given an input of abundances, we get a probability of how likely our sequence of reads is. So from a set of reads and transcripts, we can build a likelihood function and calculate the values for gamma that will maximize this function. Cufflinks achieves this using hill climbing or EM on the log-likelihood function.

11.3.4 Differential analysis with RNA-Seq

Suppose we perform an RNA-Seq analysis for a gene under two different conditions. How can we tell if there is a significant difference in the fragment counts? We calculate expression by estimating the expected number of fragments that come from each transcript. To test for significance, we need to know the variance of that estimate. We model the variance as:

$$\text{Var(expression)} = \text{Technical variability} + \text{Biological variability}$$

Technical variability, which is variability from uncertainty in mapping reads, can be modeled well with a Poisson distribution (see figure below). However, using Poisson to model biological variability, or variability across replicates, results in overdispersion.

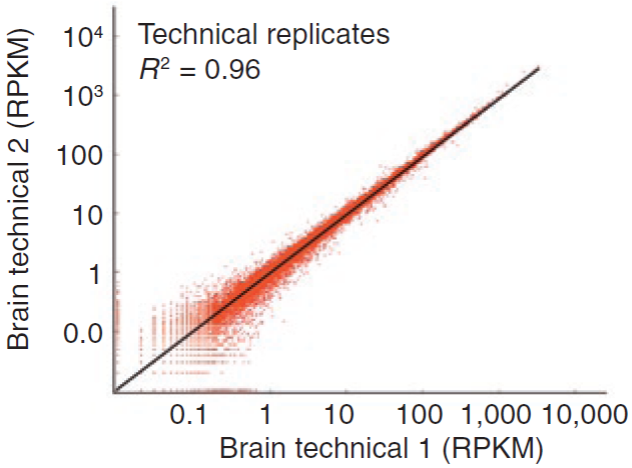


Figure 11.8: Technical variability follows a Poisson distribution

In the simple case where we have variability across replicates, but no uncertainty, we can mix the Poisson distributions from each replicate into a new distribution to model biological variability. We can treat the lambda parameter of the Poisson distribution as a random variable that follows a gamma distribution:

$$X \sim Poisson(\Gamma(r, p))$$

The counts from this model follow a negative binomial distribution. To figure out the parameters for the negative binomial for each gene, we can fit a gamma function through a scatter plot of the mean count vs. count variance across replicates.

In the simple case where there is read mapping uncertainty, but not biological variability, we need to include the mapping uncertainty in our variance estimate. Since we assign reads to transcripts probabilistically, we need to calculate the variance in that assignment.

The two threads of RNA-Seq expression analysis research focus on the problems in these two simple cases. One of the threads focuses on inferring the abundances of individual isoforms to learn about differential splicing and promoter use, while the other thread focuses on modeling variability across replicates to create more robust differential gene expression analysis. Cuffdiff unites these two separate threads to study the case where we have biological variability and read mapping ambiguity. Since overdispersion can be modeled with a negative binomial distribution and mapping uncertainty can be modeled with a Beta distribution, we combine these two to model this case with a beta negative binomial distribution.

11.4 Long non-coding RNAs in Epigenetic Regulation

Let's examine human skin as an example of long non-coding RNAs being used in epigenetic regulation. Human skin is huge, in fact it is the largest organ by weight in the body. It is intricate, with specialized features, and it is constantly regenerating to replace old dead cells with new ones. The skin must be controlled so hair only grows on the back of your hand rather than on your palm. Moreover, these boundaries cannot

change and are maintained ever since birth.

The skin in all parts of the body is composed of an epithelial layer and a layer of connective tissue made up of cells called fibroblasts. These fibroblasts secrete cytokine signals that control the outer layer, determining properties such as the presence or absence of hair. Fibroblasts all around the body are identical except for the specific epigenetic folding that dictates what type of skin will be formed in a given location. Based on whether the skin is distal or proximal, interior or exterior, posterior or anterior, a different set of epigenetic folds will determine the type of skin that forms.

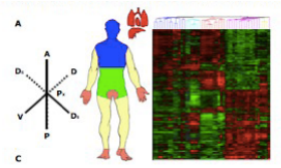


Figure 11.9: Human fibroblasts specialize via epigenetic regulation to form different skin types based on their location within the body. Research has found that the type of skin in the hands shares a remarkably similar epigenetic signature to the skin in the feet, which is also distally located.

It has been found that specific HOX genes delineate these anatomical boundaries during development. Just by looking at the human HOX genetic code, one can predict where a cell will be located. Using ChIP-on-chip (chromatin immunoprecipitation microarrays) diamteric chromatin domains have been found among these HOX genes. In the figure below, we can see a clear boundary between the chromatin domains of a cell type located proximally and another located distally. Not only is this boundary precise, but it is maintained across trillions of skin cells.

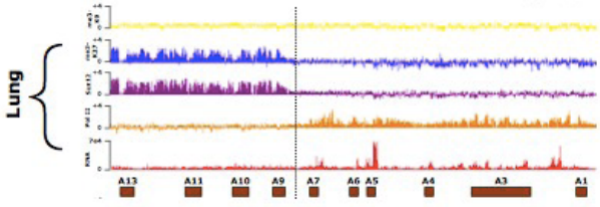


Figure 11.10: Two skin cell types are analyzed for their chromatin domains. There exists a clear boundary between the lung cell type which is proximal to the body, and the foot cell type which is distal to the body.

HOTAIR or HOX transcript antisense intergenic RNA has been investigated as possible RNA regulator that keeps these boundary between the diametric domains in chromatin. When HOTAIR was knocked out in the HOXC locus, it was hypothesized that the chromatin domains might slip through into one another. While it was found that this HOTAIR did not directly affect the epigenetic boundary, researchers did find evidence of RNA based genomic cross talk. The HOTAIR gene affected a different locus called HOXD.

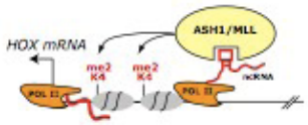


Figure 11.11: Polycomb, a protein that can remodel chromatin so that epigenetic silencing of genes can take place, may be regulated by non-coding RNA such as HOTAIR.

Through a process of ncRNA dependent Polycomb repression, the HOTAIR sequence can control epigenetic regulation. Plycomb is a portein that puts stop marks on the tails of histones so that they can cause specific folds in the genetic material. On their own histones, are undirected, so it is necessary for some

mechanism to dictate how they attach to the genome. This process of discovery has led to great interest in the power of long intergenic non-coding RNAs to affect epigenetic regulation.

11.5 Integergenic Non-coding RNAs: missing lincs in Stem/Cancer cells?

11.5.1 An example: XIST

XIST was one of the first lincRNAs to be characterized. It is directly involved in deactivation of one of the female X chromosomes during embryonic development. It has been described as having the ability to "crumple an entire chromosome". This is important because deactivation prevents lethal overexpression of genes found on the X chromosome.

RNA is important for getting polychrome complex to chromosome ncRNAs can activate downstream genes in Cis, opposite in trans; Xist does the same thing.

11.6 Technologies: in the wet lab, how can we find these?

How would we find ncRNAs? We have about 20-30 examples of ncRNAs with evidence of importance, but more are out there. Chromatin state maps (from ENCODE, chip-seq) can be used to find transcriptional units that do not overlap proteins. We can walk along map and look for genes (look by eye at chromatin map to find ncRNAs). Nearly 90% of time such a signature is found, RNA will be transcribed from it. We can validate this through northern blot

When looking at a chromatin map to find ncRNAs, we are essentially looking through the map with a window of a given size and seeing how much signal vs. noise we are getting, compared to what we might expect from a random-chance hypothesis. As both large and small windows have benefits, both should be used on each map section. Larger windows encapulate more information; smaller windows are more sensitive.

After finding integenic regions, we find conserved regions.

We check if new regions are under selective pressure; fewer mutations in conserved regions. If a nucleotide never has a mutation between species, it's highly conserved.

linc-RNAs are more conserved than introns, but less conserved than protein-coding introns, possibly due to non-conserved sequences in loop regions of lincRNAs.

Finding what lincRNAs' functions are: "Guilt by association": We can find proteins that correlate with particular lincRNA in terms of expression; lincRNAs are probably correlated to a particular pathway. In this way, we acquire a multidimensional barcode for each lincRNA (what it is and is not related to). We Can cluster lincRNA signatures and identify common patterns. Lots have to do with cell cycle genes. (This approach works 60-70% of the time)

As most lincRNAs are over 3000 bases, many contain sequences for 100 amino acid open reading frames, simply by chance. This results in many false negatives during detection.

It has been found that many lincRNAs tend to neighbor developmental regions of the genome. They also tend to be lowly expressed compared to protein coding genes.

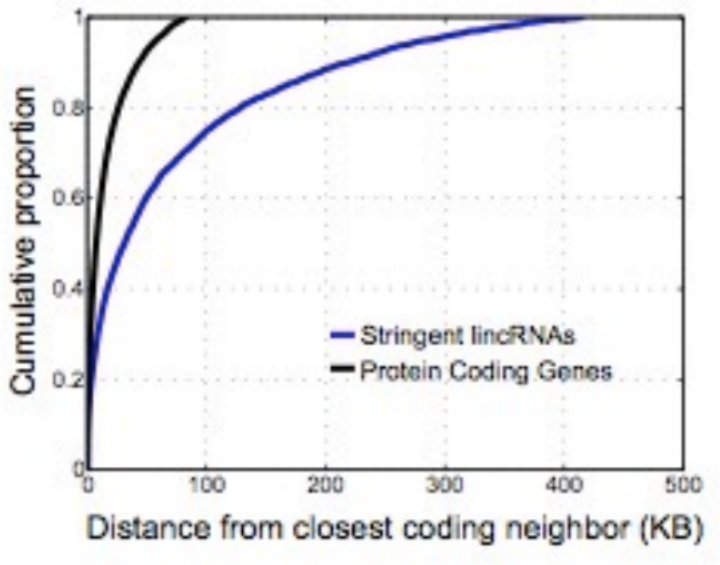


Figure 11.12: lincRNAs neighbor developmental regulators

11.6.1 Example: p53

Independent validation: we use animal models, where one is a wild-type p53, andone is a knockout. We induce p53, then ask if lincRNAs turn on. 32 of 39 lincRNAs found associated with p53 were temporally induced upon turning on p53.

One RNA in particular sat next to a protein-coding gene in the p53 pathway. We tried to figure out if p53 bound to promoter and turned it on. To do this, we cloned the promoter of lincRNA, and asked does p53 turn it on? We IPed the p53 protein, to see if it associated with the lincRNA of the promoter. It turned out that lincRNA is directly related to p53 - p53 turns it on. P53 also turns genes off - certain lincRNAs act as a repressor.

From this example (and others), we start to see that RNAs usually have a protein partner

RNA can bring myriad of different proteins together, allowing the cell lots of diversity. In this way its similar to phosphorylation. RNAs bind to important chromatin complexes, and is required for reprogramming skin cells into stem cells.

11.7 Current Research Directions

11.8 Further Reading

11.9 Tools and Techniques

11.10 What Have We Learned?

Bibliography

[1] R.P. Dilworth. A decomposition theorem for partially ordered sets. *Annal of Mathematics*, 1950.

[2] Mitchell Guttman, Manuel Garber, Joshua Z Levin, Julie Donaghey, James Robinson, Xian Adiconis, Lin Fan, Magdalena J Koziol, Andreas Gnirke, Chad Nusbaum, and et al. Ab initio reconstruction of cell type-specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincrnas. *Nature Biotechnology*, 28(5):503–510, 2010.

[3] C. Trapnell.

CHAPTER

TWELVE

SMALL RNA

Guest Lecture by David Bartel (MIT/Whitehead/HHMI) (dbartel@wi.mit.edu)
Scribed by Boyang Zhao (bozhao@mit.edu) (2011)

Figures

12.1 Introduction

Large-scale analyses in the 1990s using expressed sequence tags have estimated a total of 35,000 - 100,000 genes encoded by the human genome. However, the complete sequencing of human genome has surprisingly revealed that the numbers of protein-coding genes are likely to be ∼20,000 – 25,000 [12]. While this represents <2% of the total genome sequence, whole genome and transcriptome sequencing and tiling resolution genomic microarrays suggests that over >90% of the genome is still actively transcribed [8], largely as non-protein-coding RNAs (ncRNAs). Although initial speculation has been that these are non-functional transcriptional noise inherent in the transcription machinery, there has been rising evidence suggesting the important role these ncRNAs play in cellular processes and manifestation/progression of diseases. Hence these findings challenged the canonical view of RNA serving only as the intermediate between DNA and protein.

12.1.1 ncRNA classifications

The increasing focus on ncRNA in recent years along with the advancements in sequencing technologies (i.e. Roche 454, Illumina/Solexa, and SOLiD; refer to [16] for a more details on these methods) has led to an explosion in the identification of diverse groups of ncRNAs. Although there has not yet been a consistent nomenclature, ncRNAs can be grouped into two major classes based on transcript size: small ncRNAs (<200

nucleotides) and long ncRNAs (lncRNAs) (≥200 nucleotides) (Table 12.1[1]) [6, 8, 13, 20, 24]. Among these, the role of small ncRNAs microRNA (miRNA) and small interfering RNA (siRNA) in RNA silencing have been the most well-documented in recent history. As such, much of the discussion in the remainder of this chapter will be focused on the roles of these small ncRNAs. But first, we will briefly describe the other diverse set of ncRNAs.

Table 12.1: ncRNA classifications (based on [6, 8, 13, 20, 24])

Name	Abbreviation	Function
	Housekeeping RNAs	
Ribosomal RNA	rRNA	translation
Transfer RNA	tRNA	translation
Small nucleolar RNA	snoRNA (∼60-220 nt)	rRNA modification
Small Cajal body-specific RNA	scaRNA	splicesome modification
Small nuclear RNA	snRNA (∼60-300 nt)	RNA splicing
Guide RNA	gRNA	RNA editing
	Small ncRNAs (<200 nt)	
MicroRNA	miRNA (∼19-24 nt)	RNA silencing
Small interfering RNA	siRNA (∼21-22 nt)	RNA silencing
Piwi interacting RNA	piRNA (∼26-31 nt)	Transposon silencing, epigenetic regulation
Tiny transcription initiation RNA	tiRNA (∼17-18 nt)	Transcriptional regulation?
Promoter-associated short RNA	PASR (∼22-200 nt)	*unknown*
Transcription start site antisense RNA	TSSa-RNA (∼20-90 nt)	Transcriptional maintainence?
Termini-associated short RNA	TASR	*not clear*
Antisense termini associated short RNA	aTASR	*not clear*
Retrotransposon-derived RNA	RE-RNA	*not clear*
3'UTR-derived RNA	uaRNA	*not clear*
x-ncRNA	x-ncRNA	*not clear*
Small NF90-associated RNA	snaR	*not clear*
Unusually small RNA	usRNA	*not clear*
Vault RNA	vtRNA	*not clear*
Human Y RNA	hY RNA	*not clear*
	Long ncRNAs (≥200 nt)	
Large intergenic ncRNA	lincRNA	Epigenetics regulation
Transcribed ultraconserved regions	T-UCR	miRNA regulation?
Pseudogenes	*none*	miRNA regulation?
Promoter upstream transcripts	PROMPT	Transcriptional activation?
Telomeric repeat-containing RNA	TERRA	telomeric heterochromatin maintenance
GAA-repeat containing RNA	GRC-RNA	*not clear*
Enhancer RNA	eRNA	*not clear*
Long intronic ncRNA	*none*	*not clear*
Antisense RNA	aRNA	*not clear*
Promoter-associated long RNA	PALR	*not clear*
Stable excised intron RNA	*none*	*not clear*
Long stress-induced non-coding transcripts	LSINCT	*not clear*

[1] **TODO:** *@scribe: In Table 12.1, ncRNAs with functions labeled not clear have not yet been extensively searched in literature. There can be recent studies that suggest the functional roles of these ncRNAs.*

12.1.2 Small ncRNA

For the past decades, there have been a number of well-studied small non-coding RNA species. All of these species are either involved in RNA translation (transfer RNA (tRNA)) or RNA modification and processing (small nucleolar RNA (snoRNA) and small nuclear RNA (snRNA)). In particular, snoRNA (grouped into two broad classes: C/D Box and H/ACA Box, involved in methylation and pseudouridylation, respectively) are localized in the nucleous and participates in rRNA processing and modification. Another group of small ncRNAs are snRNAs that interact with other proteins and with each other to form splicesomes for RNA splicing. Remarkably, these snRNAs are modified (methylation and pseudouridylation) by another set of small ncRNAs - small Cajal body-specific RNAs (scaRNAs), which are similar to snoRNA (in sequence, structure, and function) and are localized in the Cajal body in the nucleus. Yet in another class of small ncRNAs, guide RNAs (gRNAs) have been shown predominately in trypanosomatids to be involved in RNA editing. Many other classes have also been recently proposed (see Table 12.1) although their functional roles remain to be determined. Perhaps the most widely studied ncRNA in the recent years are microRNAs (miRNAs), involved in gene silencing and responsible to the regulation of more than 60% protein-coding genes [6]. Given the extensive work that has been focused on RNAi and wide range of RNAi-based applications that have emerged in the past years, the next section (RNA Interference) will be entirely devoted to this topic.

12.1.3 Long ncRNA

Long ncRNAs (lncRNAs) make up the largest portion of ncRNAs [6]. However the emphasis placed on the study of long ncRNA has only been realized in the recent years. As a result, the terminology for this family of ncRNAs are still in its infancy and oftentimes inconsistent in the literature. This is also in part complicated by cases where some lncRNAs can also serve as transcripts for the generation of short RNAs. In light of these confusions, as discussed in the previous chapter, lncRNA have been arbitrarily defined as ncRNAs with size greater than 200 nts (based on the cut-off in RNA purification protocols) and can be broadly categorized into: sense, antisense, bidirectional, intronic, or intergenic [19]. For example, one particular class of lncRNA called long intergenic ncRNA (lincRNA) are found exclusively in the intergenic region and possesses chromatin modifications indicative of active transcription (e.g. H3K4me3 at the transcriptional start site and H3K36me3 throughout the gene region) [8].

Despite the recent rise of interest in lncRNAs, the discovery of the first lncRNAs (*XIST* and *H19*), based on searching cDNA libraries, dated back to the 1980s and 1990s before the discovery of miRNAs [3, 4]. Later studies demonstrated the association of lncRNAs with polycomb group proteins, suggesting potential roles of lncRNAs in epigenetic gene silencing/activation [19]. Another lncRNA, *HOX Antisense Intergenic RNA (HOTAIR)*, was recently found to be highly upregulated in metastatic breast tumors [11]. The association of *HOTAIR* with the polycomb complex again supports a potential unified role of lncRNAs in chromatin remodeling/epigenetic regulation (in either a *cis*-regulatory (*XIST* and *H19*), or *trans*-regulatory (e.g. *HOTAIR*) fashion) and disease etiology.

Recent studies have also identified *HULC* and pseudogene (transcript resembling real genes but contains mutations that prevent their translation into functional proteins) *PTENP1* that may function as a decoy in binding to miRNAs to reduce the overall effectiveness of miRNAs [18, 25]. Other potential roles of lncRNAs remains to be explored. Nevertheless, it is becoming clear that lncRNAs are less likely to be the result of transcriptional noise, but may rather serve critical role in the control of cellular processes.

12.2 RNA Interference

RNA interference has been one of the most significant and exciting discoveries in recent history. The impact of this discovery is enormous with applications ranging from knockdown and loss-of-function studies to the generation of better animal models with conditional knockdown of desired gene(s) to large-scale RNAi-based screens to aid drug discovery.

12.2.1 History of discovery

The discovery of the gene silencing phenomenon dated back as early as the 1990s with Napoli and Jorgensen demonstrating the down-regulation of chalcone synthase following introduction of exogenous transgene in plants [17]. Similar suppression was subsequently observed in other systems [10, 22]. In another set unrelated work at the time, Lee et al. identified in a genetic screen that endogenous *lin-4* expressed a non-protein-coding product that is complementary to the lin-14 gene and controlled the timing of larval development (from first to second larval state) in *C. elegans* [15]. We now know this as the first miRNA to be discovered. In 2000, another miRNA, *let-7*, was discovered in the same organism and was found to be involved in promoting the late-larval to adult transition [21]. The seminal work by Mello and Fire in 1998 (for which was awarded the Nobel Prize in 2006) demonstrated that the introduction of exogenous dsRNA in *C. elegans* specifically silenced genes via RNA interference, explaining the prior suppression phenomenon observed in plants [7]. Subsequent studies found the conversion of dsRNA into siRNA in the RNAi pathway. In 2001, the term miRNA and the link between miRNA and RNAi was described in three papers in *Science* [23]. With this, we have come to realize the gene regulatory machinery was composed of predominately of two classes small RNAs, with miRNA involved in the regulation of endogenous genes and siRNA involved in defense in response to viral nucleic acids, transposons, and transgenes [5]. Later works revealed downstream effectors: Dicers (for excision of precursor species) and Argonaute proteins (part of the RNA-induced silencing complex to perform the actual silencing effects), completing our current understanding of the RNA silencing pathways. The details of the mechanism and the differences among the species are further discussed below.

12.2.2 Biogenesis pathways

There is a common theme involved for both siRNA-mediated and miRNA-mediated silencing. In the biogenesis of both siRNA and miRNA, the double-stranded precursors are cleaved by a RNase into short ∼22 nt fragments. One of the strands (the guide strand) is loaded into an Argonaute protein, a central component of the larger ribonucleoprotien complex RISC that facilitates target RNA recognition and silencing. The mechanism of silencing are either cleaveage of the target mRNA or translation repression.

Aside from this common theme, the proteins involved in these processes differ among species and there exists additional steps in miRNA processing prior to its maturation and incorporation into RISC (Figure 12.1). For the biogenesis of siRNA, the precursors are dsRNAs, oftentimes from exogenous sources such as viruses or transposons. However, recent studies have also found endogenous siRNAs [9]. Regardless of the source, these dsRNAs are processed by the RNase III endonuclease, Dicer, into ∼22 nt siRNAs. This RNase III-catalyzed cleavage leaves the characteristic 5'phosphates and 2 nt 3' overhangs [2]. It is worth noting that different species have evolved with different number of paralogs. This becomes important as, to be discussed later, the miRNA biogenesis pathway also utilizes Dicer for the processing of miRNA precursors (more specifically pre-miRNAs). For species such as *D. melanogaster*, there are two distinct Dicer proteins and as a result there is typically a preferential processing of the precursors (e.g. Dicer-1 for miRNA cleavage and Dicer-2 for siRNA cleavage) [5]. In contrast, mammals and nematodes only have a single Dicer protein and as such both biogenesis pathways converge to the same processing step [5]. In subsequent steps of the

siRNA biogenesis pathway, one of the strands in the siRNA duplex is loaded into RISC to silence target RNAs (Figure 12.1C).

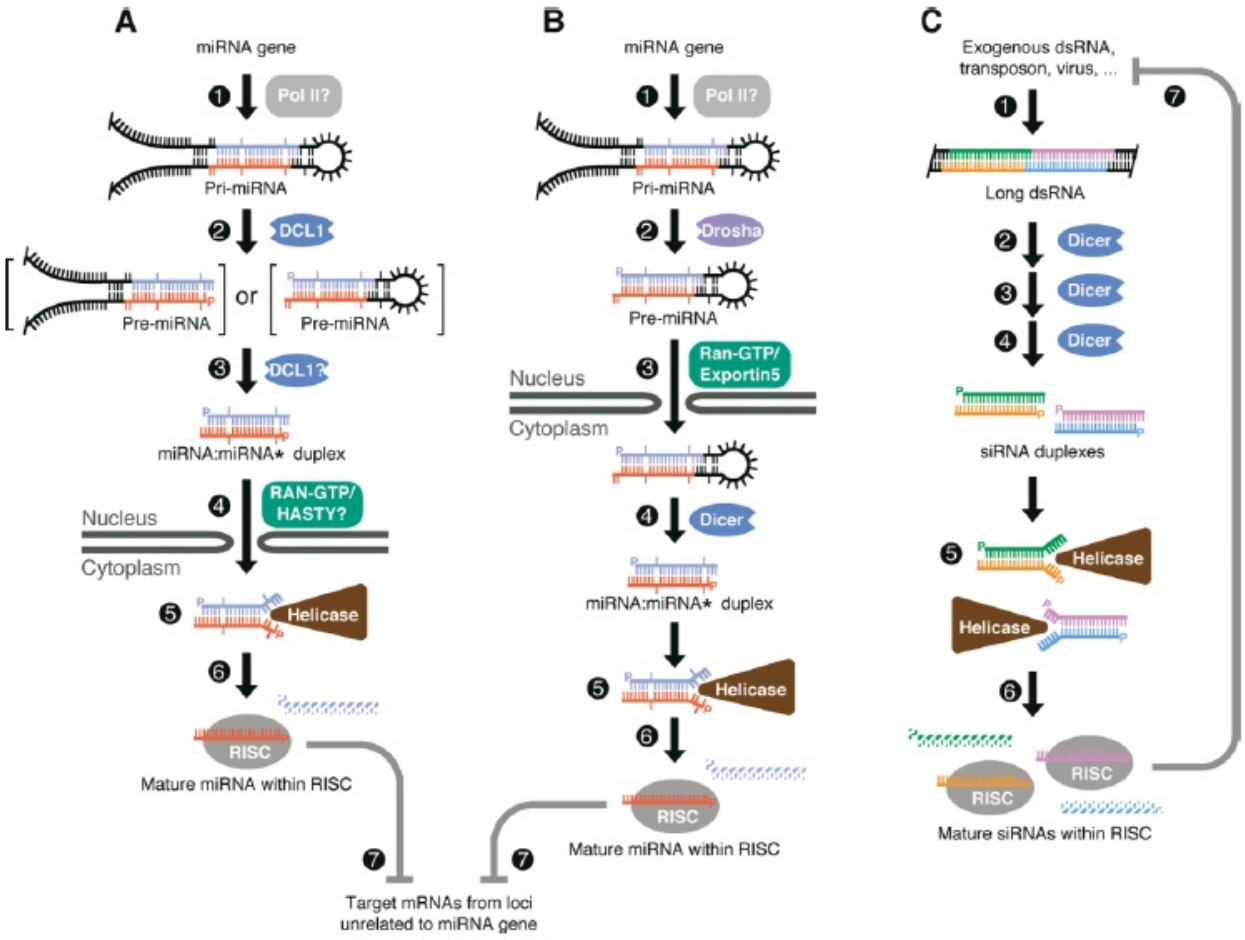


Figure 12.1: siRNA and miRNA biogenesis pathways. (A) Biogenesis of plant miRNA (B) Biogenesis of animal miRNA (C) Biogenesis of animal siRNA. Adopted from Bartel, 2004 (ref [2]). Copyright © 2004 Cell Press.

In the miRNA biogenesis pathway, majority of the precursors are pol II transcripts of the intron regions, some of which encode multiple miRNAs in clusters. These precursors, in the form of a stem-loop structure, are named pri-miRNAs. The pri-miRNAs are first cleaved in the nucleus by a RNase III endonuclease (Drosha in animals and Dcl1 in plants) into ∼60-70 nt stem loop intermediates, termed pre-miRNAs [2]. In animals, the pre-miRNA is then exported into the cytoplasm by Exportin-5. This is followed by the cleavage of pre-miRNA intermediate by Dicer to remove the stem loop. One of the strands in the resulting mature miRNA duplex is loaded to RISC, similar to that described for siRNA biogenesis Figure 12.1B. Interestingly, in plants, the pri-miRNA is processed into mature miRNA through two cleavages by the same enzyme, Dcl1, in the nucleus before export into the cytoplasm for loading (Figure 12.1A).

12.2.3 Functions and silencing mechanism

The classical view of miRNA function based on the early discoveries of miRNA has been analogous to a binary switch whereby miRNA represses translation of a few key mRNA targets to initiate a developmental

transition. However, subsequent studies have greatly broaden this definition. In plants, most miRNAs bind to the coding region of the mRNA with near-perfect complementarity. On the other hand, animal miRNAs bind with partial complementarity (except for a seed region, residues 2-8) to the 3' UTR regions of mRNA. As such, there are potentially hundreds targets by a single miRNA in animals rather than just a few [1]. In addition, in mammals, only a few portion of the predicted targets are involved in development, with the rest predicted to cover a wide range of molecular and biological processes [2]. Lastly, miRNA silencing acts through both translation repression and mRNA cleavage (and also destabilization as discussed below)(as shown for example showed by Bartel and coworkers on the miR-196-directed cleavage of *HOXB6* [26]). Taken together, the modern view of miRNA function has been that miRNA dampens expression of many mRNA targets to optimize expression, reinforce cell identity, and sharpen transitions.

The mechanism for which miRNA mediates the silencing of target mRNA is still an area of active research. As previously discussed, RNA silencing can take the form of either cleavage, destabilization (leading to subsequent degradation of the mRNA), or translation repression. In plants, it has been found that the predominate mode of RNA silencing is through Argonaute-catalyzed cleavage. However, the contribution of these different modes of silencing has been less clear in animals. Recent global analyses from the Bartel group in collaboration with Gygi and Ingolia and Weissman shed light on this question. In a 2008 study, Bartel and Gygi groups examined the global changes in protein level using mass spectrometry following miRNA introduction or deletion [1]. Their results revealed the repression of hundreds of genes by individual miRNAs, and more importantly mRNA destabilization accounts for majority of the highly repressed targets (Figure 12.2).

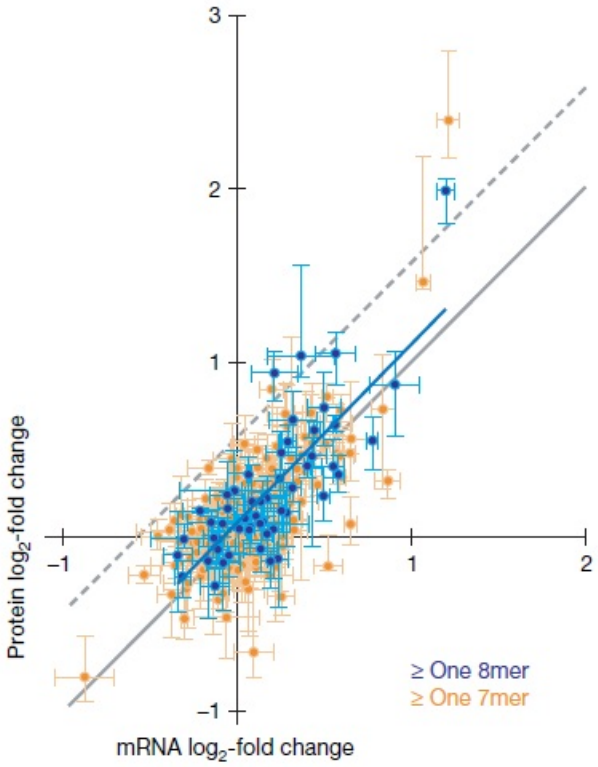


Figure 12.2: Protein and mRNA changes following miR-223 loss, from messages with at least one 8-mer 3'UTR site (blue) or at least one 7-mer (orange). Adopted from Baek et al., 2008 (ref [1]). Copyright © 2008 Macmillan Publishers Limited.

This is further supported by a subsequent study using both RNA-seq and a novel ribosome-profiling first demonstrated by Inoglia and Weissman 2009 that enables the interrogation of global translation activities

with sub-codon resolution [14]. The results showed destabilization of target mRNA is the predominate mechanism through which miRNA reduces the protein output.

Bibliography

[1] Daehyun Baek, Judit Villén, Chanseok Shin, Fernando D Camargo, Steven P Gygi, and David P Bartel. The impact of microRNAs on protein output. *Nature*, 455(7209):64–71, September 2008.

[2] David P Bartel. MicroRNAs: genomics, biogenesis, mechanism, and function. *Cell*, 116(2):281–97, January 2004.

[3] M S Bartolomei, S Zemel, and S M Tilghman. Parental imprinting of the mouse H19 gene. *Nature*, 351(6322):153–5, May 1991.

[4] C J Brown, A Ballabio, J L Rupert, R G Lafreniere, M Grompe, R Tonlorenzi, and H F Willard. A gene from the region of the human X inactivation centre is expressed exclusively from the inactive X chromosome. *Nature*, 349(6304):38–44, January 1991.

[5] Richard W Carthew and Erik J Sontheimer. Origins and Mechanisms of miRNAs and siRNAs. *Cell*, 136(4):642–55, February 2009.

[6] Manel Esteller. Non-coding RNAs in human disease. *Nature Reviews Genetics*, 12(12):861–874, November 2011.

[7] A Fire, S Xu, M K Montgomery, S A Kostas, S E Driver, and C C Mello. Potent and specific genetic interference by double-stranded RNA in Caenorhabditis elegans. *Nature*, 391(6669):806–11, February 1998.

[8] Ewan a Gibb, Carolyn J Brown, and Wan L Lam. The functional role of long non-coding RNA in human carcinomas. *Molecular cancer*, 10(1):38, January 2011.

[9] Daniel E Golden, Vincent R Gerbasi, and Erik J Sontheimer. An inside job for siRNAs. *Molecular cell*, 31(3):309–12, August 2008.

[10] S Guo and K J Kemphues. par-1, a gene required for establishing polarity in C. elegans embryos, encodes a putative Ser/Thr kinase that is asymmetrically distributed. *Cell*, 81(4):611–20, May 1995.

[11] Rajnish A Gupta, Nilay Shah, Kevin C Wang, Jeewon Kim, Hugo M Horlings, David J Wong, Miao-Chih Tsai, Tiffany Hung, Pedram Argani, John L Rinn, Yulei Wang, Pius Brzoska, Benjamin Kong, Rui Li, Robert B West, Marc J van de Vijver, Saraswati Sukumar, and Howard Y Chang. Long non-coding RNA HOTAIR reprograms chromatin state to promote cancer metastasis. *Nature*, 464(7291):1071–6, April 2010.

[12] Masahira Hattori. Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–45, October 2004.

[13] Christopher L Holley and Veli K Topkara. An introduction to small non-coding RNAs: miRNA and snoRNA. *Cardiovascular Drugs and Therapy*, 25(2):151–159, 2011.

[14] Nicholas T Ingolia, Sina Ghaemmaghami, John R S Newman, and Jonathan S Weissman. Genome-wide analysis in vivo of translation with nucleotide resolution using ribosome profiling. *Science (New York, N.Y.)*, 324(5924):218–23, April 2009.

[15] R C Lee, R L Feinbaum, and V Ambros. The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. *Cell*, 75(5):843–54, December 1993.

[16] Michael L Metzker. Sequencing technologies - the next generation. *Nature Reviews Genetics*, 11(1):31–46, January 2010.

[17] C. Napoli, C. Lemieux, and R. Jorgensen. Introduction of a Chimeric Chalcone Synthase Gene into Petunia Results in Reversible Co-Suppression of Homologous Genes in trans. *The Plant cell*, 2(4):279–289, April 1990.

[18] Laura Poliseno, Leonardo Salmena, Jiangwen Zhang, Brett Carver, William J Haveman, and Pier Paolo Pandolfi. A coding-independent function of gene and pseudogene mRNAs regulates tumour biology. *Nature*, 465(7301):1033–8, June 2010.

[19] Chris P Ponting, Peter L Oliver, and Wolf Reik. Evolution and functions of long noncoding RNAs. *Cell*, 136(4):629–41, February 2009.

[20] J. R. Prensner and A. M. Chinnaiyan. The Emergence of lncRNAs in Cancer Biology. *Cancer Discovery*, 1(5):391–407, October 2011.

[21] B J Reinhart, F J Slack, M Basson, A E Pasquinelli, J C Bettinger, A E Rougvie, H R Horvitz, and G Ruvkun. The 21-nucleotide let-7 RNA regulates developmental timing in Caenorhabditis elegans. *Nature*, 403(6772):901–6, February 2000.

[22] N Romano and G Macino. Quelling: transient inactivation of gene expression in Neurospora crassa by transformation with homologous sequences. *Molecular microbiology*, 6(22):3343–53, November 1992.

[23] G Ruvkun. Molecular biology. Glimpses of a tiny RNA world. *Science*, 294(5543):797–9, October 2001.

[24] Ryan J Taft, Ken C Pang, Timothy R Mercer, Marcel Dinger, and John S Mattick. Non-coding RNAs: regulators of disease. *The Journal of pathology*, 220(2):126–39, January 2010.

[25] Jiayi Wang, Xiangfan Liu, Huacheng Wu, Peihua Ni, Zhidong Gu, Yongxia Qiao, Ning Chen, Fenyong Sun, and Qishi Fan. CREB up-regulates long non-coding RNA, HULC expression through interaction with microRNA-372 in liver cancer. *Nucleic acids research*, 38(16):5366–83, September 2010.

[26] Soraya Yekta, I-Hung Shih, and David P Bartel. MicroRNA-directed cleavage of HOXB8 mRNA. *Science*, 304(5670):594–6, April 2004.

Part III

Gene and Genome Regulation

CHAPTER

THIRTEEN

MRNA SEQUENCING FOR EXPRESSION ANALYSIS AND TRANSCRIPT DISCOVERY

Guest lecture by Manuel Garber

Figures

13.1 Introduction

The purpose of mRNA sequencing is to measure the levels of mRNA transcripts for every gene in a given cell.

13.2 Expression Microarrays

Prior to the development of mRNA sequencing technology, mRNA levels were measured using expression microarrays. These microarrays function by inserting a DNA probe on a slide and measuring the levels transcripts that undergo complimentary hybridization with the DNA, a process that could analyze expression on a gene by gene basis (Figure 1).

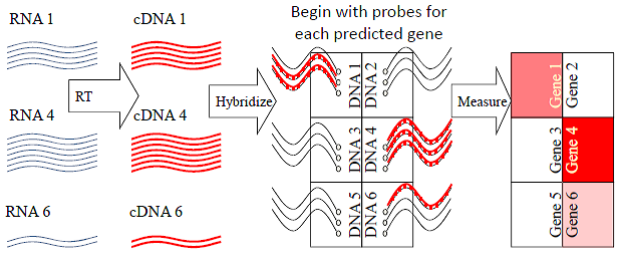


Figure 13.1: Figure 1: Expression microarray process

However, this technology has several limitations: it cannot distinguish mRNA isoforms, it cannot analyze on the sequence, or digital level, it can only measure known transcripts, and the expression measurements become less reliable for highly saturated transcript levels.

mRNA sequencing was a daunting task, and would require approximately 40 million aligned reads in order to accurately measure mRNA transcripts. This did not become possible until 2009, when next-generation sequencing technologies became more advanced and efficient.

13.3 The Biology of mRNA Sequencing

The first step in mRNA sequencing is to lyse the cells of interest. This creates a mass of proteins, nucleotides, and other molecules, which must be filtered out so that only RNA (or specifically mRNA) molecules remain. The resulting transcripts are then fragmented into reads 200-1000 base pairs long and undergo a reverse transcription reaction to build a strand-specific DNA library. Finally, both ends of these DNA fragments are sequenced. After establishing these sequenced reads, the computational part of RNA-Seq can be divided into three parts: read mapping, reconstruction, and quantification.

13.4 Read Mapping - Spaced Seed Alignment

The idea behind read mapping is to align the sequenced reads to a reference genome. This process begins by using the reference genome to creating a hash table of 8-mers, which do not have to be contiguous. The positions of these stored spaced seeds are mapped to the hash table. Using these spaced 8-mers, each read is then compared with each possible position in the reference genome and scored based on the number of base pair matches (Figure 2).

More accurately, for each position, it is possible to calculate the score using the equation $q_{MS} = -10\log_{10}(1 - P(i|G,q))$, where $P(i|G,q)$ represents the probability that the read, q, is mapped to position i of reference genome G. More details on deriving this score can be found in Box 1.

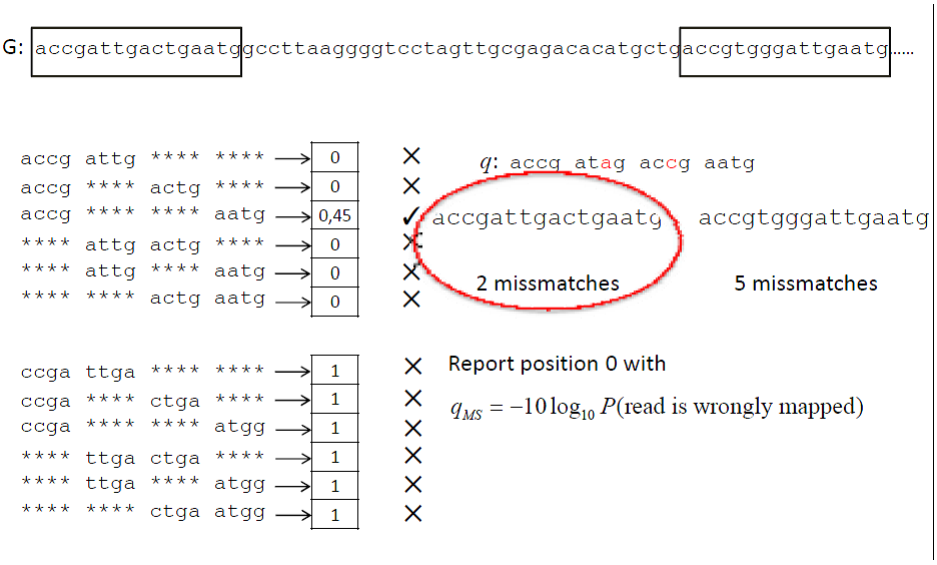


Figure 13.2: Figure 2: Spaced k-mer method of mapping reads to reference genome

It is possible to adjust the parameters of this method in order to alter the sensitivity, speed, and memory of the algorithm. Using smaller k-mer seeds allows for less precise base pair matching (greater sensitivity), but requires more matches to be attempted. Smaller seeds also take up less memory, while larger seeds run faster. The Burrows-Wheeler transform is an even more efficient algorithm for mapping reads, and will be discussed in a later chapter. As better sequencing technology allows for larger read lengths, more algorithms will need to be developed to handle the extra processing.

Unlike ChIP-Seq, a similar technology, RNA-seq is more complex, as the read mapper needs to worry about small exons interspersed between large introns, and be able to find both sides of an exon. This complexity can be overcome by using the above mentioned spaced seed matching technique, and then extending the k-mers to fill in gaps (SNO methods). Another method is to base the alignment on contiguous reads, which are further fragmented into 20-30 bp regions. These regions are remapped, and the positions with two or more different alignments are marked as splice junctions. Exon-first aligners are faster than the previous methods, but come at a cost: they fail to differentiate psuedogenes, prespliced genes, and transposed genes.

13.5 Reconstruction

Reconstruction of reads is a largely statistical problem. The goal is to determine a score for each fixed-sized window in the genome. This score represents the probability of seeing the observed number of reads given the window size. In other words, is the number of reads in a particular window unlikely given the genome? The expected number of reads per window is derived from a uniform distribution based on the total number of reads (Figure 3). This score is modeled by a Poisson distribution.

However, this score must account for the problem of multiple testing hypotheses, due to the approximately 150 million expected bases. One option for dealing with this is the Bonferroni correction, where the nominal p-value = n * p-value. This method leads to low sensitivity, due to its very conservative nature. Another option is to permute the reads observed in the genome, and find the maximum number of reads seen on a single base. This allows for a max count distribution model, but the process is very slow. The scan distribution speeds up this process by computing a closed form for max count distribution to account for dependency of overlapping windows (Figure 4). The probability of observing k reads on a window of size w

Figure 13.3: Box 1: How Do We Calculate qMS?

What does $q_{MS} = -10\log_{10} P(\text{read is wrongly mapped})$ mean?

Lets compute the probability the read originated at genome position i

q: accg atag accg aatg

q_s: 30 40 25 30 30 20 10 20 40 30 20 30 40 40 30 25

$q_s[k] = -10\log_{10} P(\text{sequencing error at base k})$, the PHRED score. Equivalently:

$P(\text{sequencing error at base k}) = 10^{-\frac{q_s}{10}}$

So the probability that a read originates from a given genome position i is:

$$P(q \mid G,i) = \prod_{j \text{ match}} P(q_j \text{good call}) \prod_{j \text{ missmatch}} P(q_j \text{bad call}) \approx \prod_{j \text{ missmatch}} P(q_j \text{bad call})$$

In our example

$$P(q \mid G,0) = \left[(1-10^{-3})^6 (1-10^{-4})^4 (1-10^{-2.5})^2 (1-10^{-2})^2\right]\left[10^{-1}10^{-2}\right] = [0.97]*[0.001] \approx 0.001$$

What does $q_{MS} = -10\log_{10} P(\text{read is wrongly mapped})$ mean?

$$P(q \mid G,i) = \prod_{j \text{ match}} P(q_j \text{good call}) \prod_{j \text{ missmatch}} P(q_j \text{bad call}) \approx \prod_{j \text{ missmatch}} P(q_j \text{bad call})$$

But what we need is the posterior probability, the probability that the region starting at i was sequenced *given* that we observed the read q:

$$P(i \mid G,q) = \frac{P(q \mid G,i)P(i \mid G)}{P(q \mid G)} = \frac{P(q \mid G,i)P(i \mid G)}{\sum_j P(q \mid G,j)}$$

Fortunately, there are efficient ways to approximate this probability (see Li, H *genome Research* 2008, for example)

$$q_{MS} = -10\log_{10}(1 - P(i \mid G,q))$$

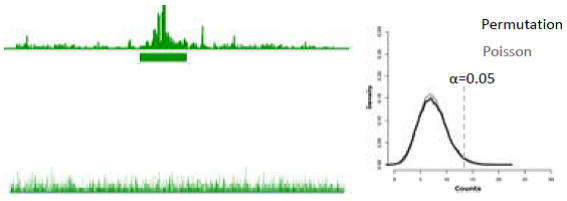


Figure 13.4: Figure 3: Reconstruction works by determining, for a particular window, the probability of observing that number of reads (top left) given the uniform distribution of the total reads (bottom left). This probability follows the Poisson distribution.

in a genome of size L given a total of N reads can be approximated by [slide is not clear].

Choosing a window size is also an important decision, as genes exist at different expression levels and span different orders of magnitude. Small windows are better at detecting punctuate regions, while larger windows can detect longer spans of moderate enhancement. In most cases, windows of different sizes are used to pick up signals of varying size.

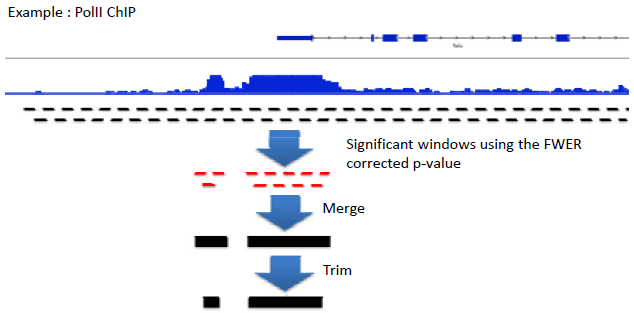


Figure 13.5: Figure 4: Process for reconstructing genome based on reads, using the scan distribution

Transcript reconstruction can be seen as a segmentation problem, with several challenges. As mentioned above, genes are expressed at different levels, over several orders of magnitude. In addition, the reads used for reconstruction are obtained from both mature and immature mRNA, the latter still containing introns. Finally, many genes have multiple isoforms, and the short nature of reads makes it difficult to differentiate between these different transcripts. A computational tool called Scripture uses a priori knowledge of fragment connectivity to detect transcripts.

Alternative isoforms can only be detected via exon junction spanning reads, which contain the ends of an exon. Longer reads have a greater chance of spanning these junctions (Figure 5). Scripture works by modeling the reads using graph structure, where bases are connected to neighbor bases, as well as splice neighbors. This process differs from the string graph technique, because it focuses on whole genome, and does not map overlapping sequences directly. When sliding the window, Scripture can jump across splice junctions yet still examine alternative isoforms. From this oriented connectivity graph, the program identifies segments across the graph, and looks for significant segments (Box 2).

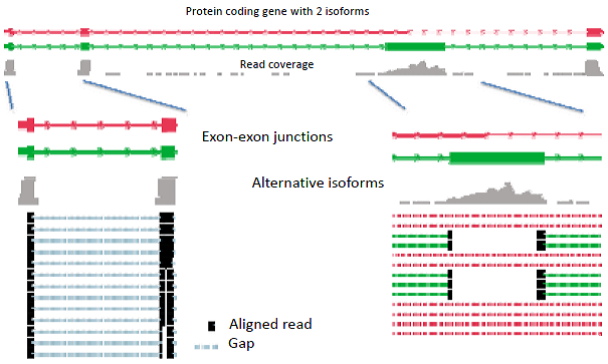
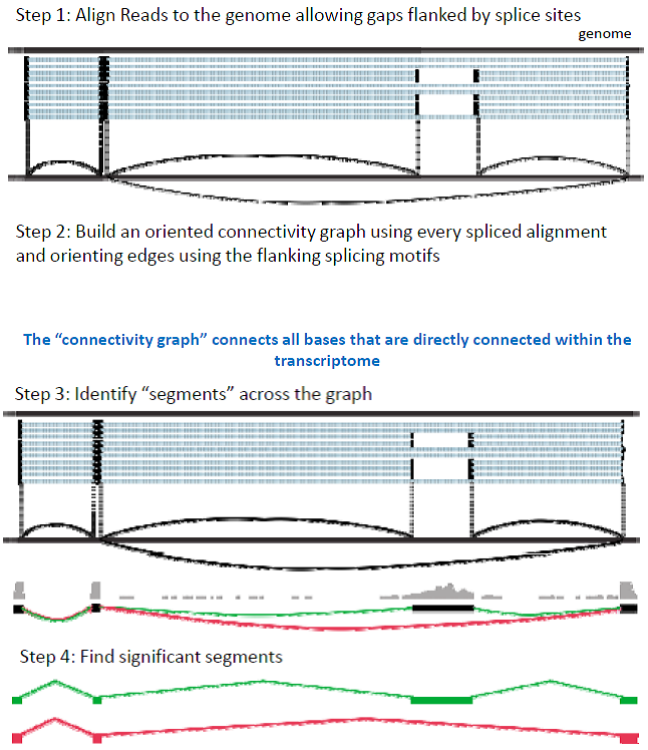


Figure 13.6: Figure 5: Alternative isoforms present a challenge for reconstruction, which must depend on exon junction spanning reads

Direct transcript assembly is another method of reconstruction (as opposed to genome-guided methods like Scripture). Transcript assembly methods are able to reconstruct transcripts from organisms without a reference sequence, while genome-guided approaches are ideal for annotating high quality genomes and expanding the catalog of expressed transcripts. Hybrid approaches are used for lesser quality transcripts or transcriptomes that have underwent major rearrangements, such as those of cancer cells. Popular transcript assembly tools include Oasis, Trans-ABySS, and Trinity. Another popular genome-guided software is Cufflinks. Regardless of methodology or software type, any sequencing experiment that produces more genome coverage will experience better transcript reconstruction.

Figure 13.7: Box 2: The Scripture Method



13.6 Quantification

The goal of the quantification step is to score regions in the genome based on the number of reads. However, it is insufficient to simply count the number of reads per region, as this number is dependent on the expression of the gene transcript: low expression transcripts can look like short transcripts, and highly expressed transcripts can resemble long transcripts. This issue can be solved by normalizing the number of reads by the length of the transcript and the total number of reads in the experiment. This provides the RPKM value, or reads per kilobase of exonic sequence per million mapped reads.

This method is robust for genes with only one isoform. When multiple transcript variants are involved, there are a few different methods for handling this complexity. The exon intersection model scores only the constituent exons. The exon union model simply scores based on a merged transcript, but can easily be biased based on the relative ratios of each isoform. A more thorough model is the transcript expression model, which assigns unique reads to different isoforms.

CHAPTER

FOURTEEN

GENE REGULATION 1 –GENE EXPRESSION CLUSTERING

Franck Dernoncourt (2012)
Arvind Thiagarajan (2011)
Tahin Syed (2010)
Barrett Steinberg and Brianna Petrone (2009)
Mia Y. Qia (2008)

Figures

14.1 Introduction

In this chapter, we consider the problem of discerning similarities or patterns within large datasets. Finding structure in such data sets allows us to draw conclusions about the process as well as the structure underlying the observations. We approach this problem through the application of clustering techniques. The following chapter will focus on classification techniques.

14.1.1 Clustering vs Classification

One important distinction to be made early on is the difference between classification and clustering. **Classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations or instances whose category membership is known. The training set is used to learn rules that will accurately assign labels to new observations. The difficulty is to find the most important features (feature selection).

In the terminology of machine learning, classification is considered an instance of supervised learning, i.e. learning where a training set of correctly-identified observations is available. The corresponding unsupervised procedure is known as **clustering** or cluster analysis, and involves grouping data into categories based on some measure of inherent similarity, such as the distance between instances, considered as vectors in a multi-dimensional vector space. The difficulty is to identify the structure of the data. Figure 14.1 illustrates the difference between clustering and classification.

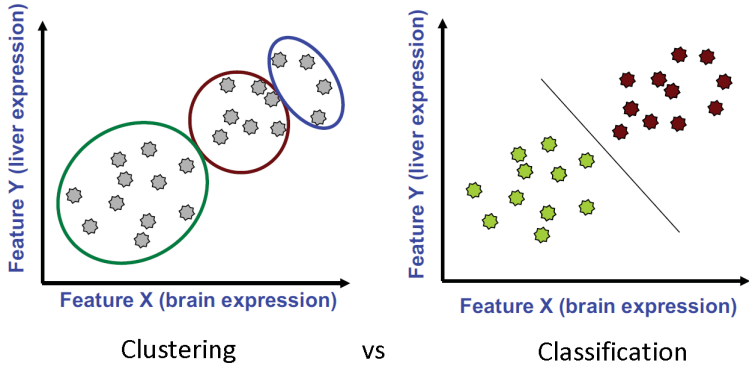


Figure 14.1: Clustering compared to classification. In clustering we group observations into clusters based on how near they are to one another. In classification we want a rule that will accurately assign labels to new points.

14.1.2 Applications

Clustering was originally developed within the field of artificial intelligence. Being able to group similar objects, with full implications of generality implied, is indeed a fairly desirable attribute for an artificial intelligence, and one that humans perform routinely throughout life. As the development of clustering algorithms proceeded apace, it quickly becomes clear that there was no intrinsic barrier involved in applying

these algorithms to larger and larger datasets. This realization led to the rapid introduction of clustering to computational biology and other fields dealing with large datasets.

Clustering has many applications to computational biology. For example, let's consider expression profiles of many genes taken at various developmental stages. Clustering may show that certain sets of genes line up (i.e. show the same expression levels) at various stages. This may indicate that this set of genes has common expression or regulation and we can use this to infer similar function. Furthermore, if we find a uncharacterized gene in such a set of genes, we can reason that the uncharacterized gene also has a similar function through guilt by association.

Chromatin marks and regulatory motifs can be used to predict logical relationships between regulators and target genes in a similar manner. This sort of analysis enables the construction of models that allow us to predict gene expression. These models can be used to modify the regulatory properties of a particular gene, predict how a disease state arose, or aid in targeting genes to particular organs based on regulatory circuits in the cells of the relevant organ.

Computational biology deals with increasingly large and open-access datasets. One such example is the ENCODE project [2]. Launched is 2003, the goal of ENCODE is to build a comprehensive list of functional elements in the human genome, including elements that act at the protein and RNA levels, and regulatory elements that control cells and circumstances in which a gene is active. ENCODE data are now freely and immediately available for the entire human genome: http://genome.ucsc.edu/ENCODE/. Using all of this data, it is possible to make functional predictions about genes through the use of clustering.

14.2 Microarrays

The most intuitive way to investigate a certain phenotype is to measure the functional proteins present at a given time in the cell. However, given the difficulty of measuring protein, due to their varying locations, modifications, and contexts in which they are found, as well as due to the incompleteness of the proteome, mRNA expression levels are often used as a good approximation, since it is much easier to measure and allows thousands of genes to be measured in one **microarray** experiment. Furthermore, given the Central Dogma of Molecular Biology, it is more desirable to measure mRNA since we are measuring the regulation that occurs at the level of the genome. By measuring proteins, we would be combining two regulatory steps.

14.2.1 Technology/Biological Methods

The basic principle behind microarrays is the hybridization of complementary DNA fragments. To begin, short segments of DNA, known as probes, are attached to a solid surface, commonly known as a gene chip. Then, the RNA population of interest, which has been taken from a cell, is reverse transcribed to cDNA (complementary DNA) via reverse transcriptase, which synthesizes DNA from RNA using the poly-A tail as a primer. For intergenic sequences which have no poly-A tail, a standard primer can be ligated to the ends of the mRNA. The resulting DNA has more complementarity to the DNA on the slide than the RNA. The cDNA is than washed over the chip and the resulting hybridization triggers the probes to fluoresce. This can be detected to determine the relative abundance of the mRNA in the target, as illustrated in figure 14.2.

Two basic types of microarrays are currently used. Affymetrix gene chips have one spot for every gene and have longer probes on the order of 100s of nucleotides. On the other hand, spotted oligonucleotide arrays tile genes and have shorter probes around the tens of bases.

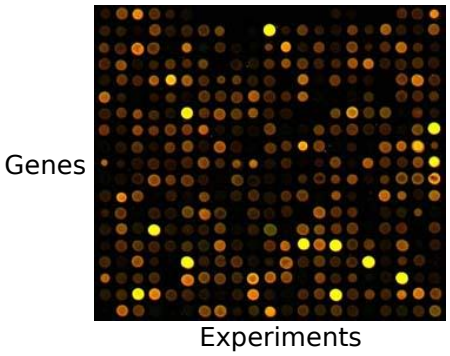


Figure 14.2: Gene expression values from microarray experiments can be represented as heat maps to visualize the result of data analysis.

14.2.2 Limits

There are numerous sources of error in the current methods and future methods seek to remove steps in the process. For instance, reverse transcriptase may introduce mismatches, which weaken interaction with the correct probe or cause cross hybridization, or binding to multiple probes. One solution to this has been to use multiple probes per gene, as cross hybridization will be different for each gene. Still, reverse transcription is necessary due to the secondary structure of RNA. The structural stability of DNA makes it less probable to bend and not hybridize to the probe. The next generation of technologies, such as RNA-Seq, sequences the RNA as it comes out of the cell, essentially probing every base of the genome.

14.3 RNA-Seq

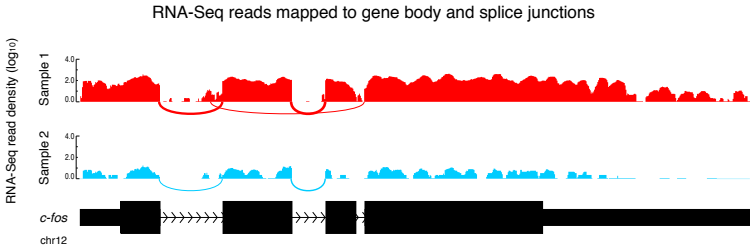


Figure 14.3: RNA-Seq reads mapping to a gene (*c-fos*) and its splice junctions. Densities along exon represent read densities mapping to exons (in $\log_{10}$), arcs correspond to junction reads, where arc width is drawn in proportion to number of reads in that junction. The gene is downregulated in Sample 2 compared to Sample 1.

RNA-Seq, also known as whole transcriptome shotgun sequencing, attempts to perform the same function that DNA microarrays have been used to perform in the past, but with greater resolution. In particular, DNA microarrays utilize specific probes, and creation of these probes necessarily depends on prior knowledge of the genome and the size of the array being produced. RNA-seq removes these limitations by simply sequencing all of the cDNA produced in microarray experiments. This is made possible by next-generation sequencing technology. The technique has been rapidly adopted in studies of diseases like cancer [4]. The data from RNA-seq is then analyzed by clustering in the same manner as data from microarrays would normally be analyzed.

14.4 Gene Expression Matrices

The amount of data generated by a microarray or an RNA-seq is enormous. For example, taking the output of a gene prediction model and making probes for every single one, detects the expression of every gene in the cell. Furthermore, performing experiments across conditions, time courses, stages of development, phenotype, or any other factor increases the amount of data. The result is often represented as an m x n expression matrix, profiling the expression levels of m genes across n experiments. Clustering gene expression data enables data visualization to identify what different cancers look like, find functional similarities between genes, and can facilitate the development of a gene regulatory network model.

These matrices can be clustered hierarchically showing the relation between pairs of genes, pairs of pairs, and so on, creating a dendrogram in which the rows and columns can be ordered using optimal leaf ordering algorithms.

This predictive and analytical power is increased due to the ability of biclustering the data; that is, clustering along both dimensions of the matrix. The matrix allows for the comparison of expression profiles of genes, as well as comparing the similarity of different conditions such as diseases. A challenge, though, is the curse of dimensionality. As the space of the data increases, the clustering of the points diminishes. Sometimes, the data can be reduced to lower dimensional spaces to find structure in the data using clustering to infer which points belong together based on proximity.

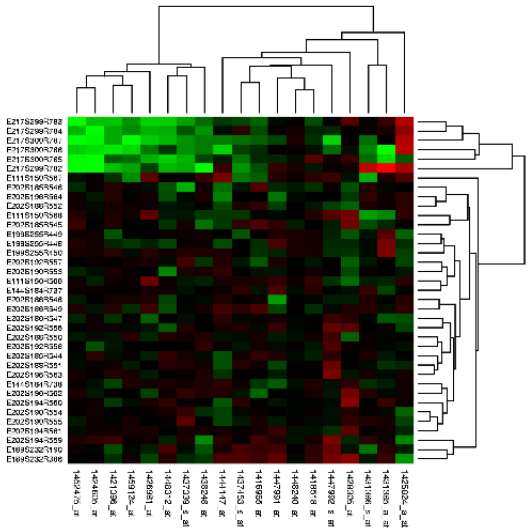


Figure 14.4: A sample matrix of gene expression values, represented as a heatmap and with hierarchal clusters. [1]

Interpreting the data can also be a challenge, since there may be other biological phenomena in play. For example, protein-coding exons have higher intensity, due to the fact that introns are rapidly degraded. At the same time, not all introns are junk and there may be ambiguities in alternative splicing. There are also cellular mechanisms that degrade aberrant transcripts through non-sense mediated decay.

14.5 Clustering Algorithms

There are two types of clustering algorithms: partitioning and agglomerative. Partitional clustering divides objects into non-overlapping clusters so that each data object is in one subset. Alternatively, agglomerative clustering methods yield a set of nested clusters organized as a hierarchy representing structures from broader to finer levels of detail.

14.5.1 *K*-Means Clustering

The k-means algorithm clusters n objects based on their attributes into k partitions. This is an example of partitioning, where each point is assigned to exactly one cluster such that the sum of distances from each point to its correspondingly labeled center is minimized. The motivation underlying this process is to make the most compact clusters possible, usually in terms of a Euclidean distance metric.

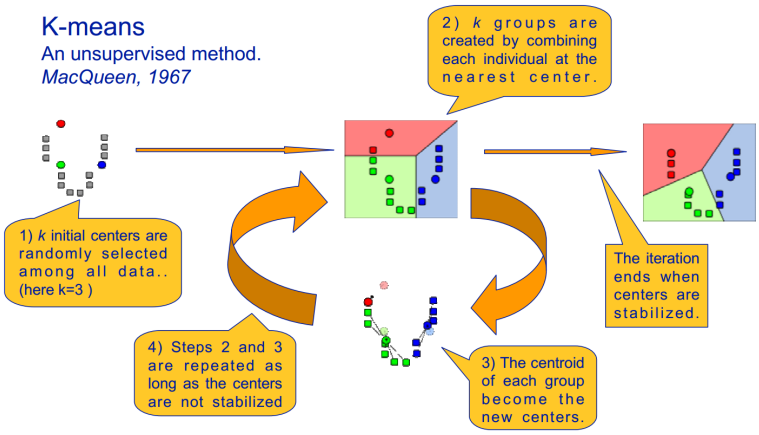


Figure 14.5: The k-means clustering algorithm

The k-means algorithm, as illustrated in figure 14.5, is implemented as follows:

1. Assume a fixed number of clusters, k
2. *Initialization*: Randomly initialize the k means μ_k associated with the clusters and assign each data point x_i to the nearest cluster, where the distance between x_i and μ_k is given by $d_{i,k} = (x_i - \mu_k)^2$.
3. *Iteration*: Recalculate the centroid of the cluster given the points assigned to it: $\mu_k(n+1) = \sum_{x_i \in k} \frac{x_i}{|x^k|}$ where x_k is the number of points with label k. Reassign data points to the k new centroids by the given distance metric. The new centers are effectively calculated to be the average of the points assigned to each cluster.
4. *Termination*: Iterate until convergence or until a user-specified number of iterations has been reached. Note that the iteration may be trapped at some local optima.

There are several methods for choosing k: simply looking at the data to identify potential clusters or iteratively trying values for n, while penalizing model complexity. We can always make better clusters by increasing k, but at some point we begin overfitting the data.

We can also think of k-means as trying to minimize a cost criterion associated with the size of each cluster, where the cost increases as the clusters get less compact. However, some points can be almost halfway between two centers, which doesn't fit well with the binary belonging k-means clustering.

14.5.2 Fuzzy K-Means Clustering

In fuzzy clustering, each point has a probability of belonging to each cluster, rather than completely belonging to just one cluster. Fuzzy k-means specifically tries to deal with the problem where points are somewhat in between centers or otherwise ambiguous by replacing distance with probability, which of course could be some function of distance, such as having probability relative to the inverse of the distance. Fuzzy k-means uses a weighted centroid based on those probabilities. Processes of initialization, iteration, and termination are the same as the ones used in k-means. The resulting clusters are best analyzed as probabilistic distributions rather than a hard assignment of labels. One should realize that k-means is a special case of fuzzy k-means when the probability function used is simply 1 if the data point is closest to a centroid and 0 otherwise.

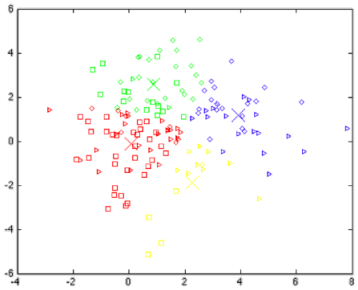


Figure 14.6: Examples of final cluster assignments of fuzzy k-means using k= 4 with centroids, correct clusters, and most probable assigned clusters marked as crosses, shapes shapes of points, and colors respectively. Note that the original data set is non-Gaussian.

The fuzzy k-means algorithm is the following:

1. Assume a fixed number of clusters k
2. *Initialization*: Randomly initialize the k means μ_k associated with the clusters and compute the probability that each data point x_i is a member of a given cluster k, $P(point x_i has label k|x_i, k)$.
3. *Iteration*: Recalculate the centroid of the cluster as the weighted centroid given the probabilities of membership of all data points x_i:
$$\mu_k(n+1) = \frac{\sum_{x_i \in k} x_i \times P\left(\mu_k|x_i\right)^b}{\sum_{x_i \in k} P\left(\mu_k|x_i\right)^b}$$
4. *Termination*: Iterate until convergence or until a user-specified number of iterations has been reached (the iteration may be trapped at some local maxima or minima)

14.5.3 K-Means as a Generative Model

A generative model is a model for randomly generating observed data, given some hidden parameters. While a generative model is a probability model of all variables, a discriminative model provides a conditional model

only of the target variable(s) using the observed variables.

When we perform clustering, we are assuming something about the underlying data. In the case of k-means and fuzzy k-means, we are assuming that a set k centers (parameters) generate the data points using a Gaussian distribution for each k, potentially generating a stochastic representation of the data, as shown in figure 14.7. In the case where $k = 2$, this can be thought of as flipping a coin to choose one of the two centers, then randomly placing a point, according to a Gaussian distribution, somewhere near the center. Unfortunately, since k-means assumes independence between the axes, covariance and variance are not accounted for using k-means, so models such as oblong distributions are not possible. In the clustering processes discussed containing a set of labeled data points x, we want to choose the most probable center (parameter) for x; that is, we want to maximize the probability of the clustering model. This is the maximum likelihood setting of μ_k, given the data. Given a set of observations x_i and all labels k, we can find the maximum likelihood μ_k as follows:

$$\arg\max_{\mu}\left\{log\prod_{i}P\left(x_i|\mu\right)\right\} = \arg\max_{\mu}\sum_{i}\left\{-\tfrac{1}{2}(x_i-\mu)^2 + log(\tfrac{1}{\sqrt{2\pi}})\right\} = \arg\min_{\mu}\sum_{i}(x_i-\mu)$$

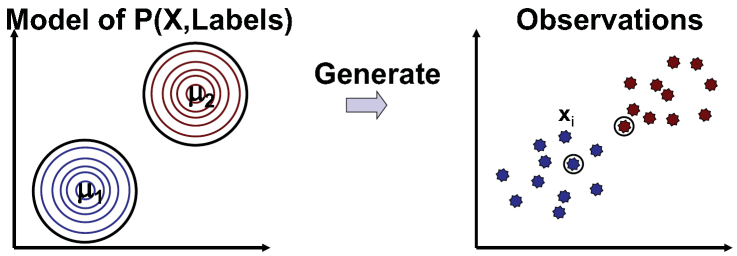


Figure 14.7: K-Means as a Generative Model. Samples were drawn from normal distributions.

Probability in this case is a function of distance of each data point from each center. After that, we want to find the best new parameter, the maximum likelihood for the next iteration of the algorithm using the same processes.

These same principles apply in reverse to determine labels from known centers, similarly using the argmin function. In this case, we attempt to find the best label that maximizes the likelihood of the data. We find that this is the same as simply finding the nearest center.

If neither labels nor centers are known, a common solution to estimate both is to start with k arbitrary centers, calculate the most likely labels given the centers, use these labels to choose new centers, and iterate until a local maximum of probability is reached.

K-means can be seen as an example of EM (expectation maximization algorithms), as shown in figure 14.8 where expectation consists of estimation of hidden labels, Q, and maximizing of expected likelihood occurs given data and Q. Assigning each point the label of the nearest center corresponds to the E step of estimating the most likely label given the previous parameter. Then, using the data produced in the E step as observation, moving the centroid to the average of the labels assigned to that center corresponds to the M step of maximizing the likelihood of the center given the labels. This case is analogous to Viterbi learning. A similar comparison can be drawn for fuzzy k-means, which is analogous to Baum-Welch from HMMs. Figure 14.9 compares clustering, HMM and motif discovery with respect to expectation minimization algorithm.

EM is guaranteed to converge and guaranteed to find the best possible answer, at least from an algorithmic point of view. The notable problem with this solution is that of local maxima of probability distributions in which only uphill movement is possible. Thus an absolute maximum may never be determined. This problem may be hopefully avoided by attempting multiple initializations to better determine the landscape

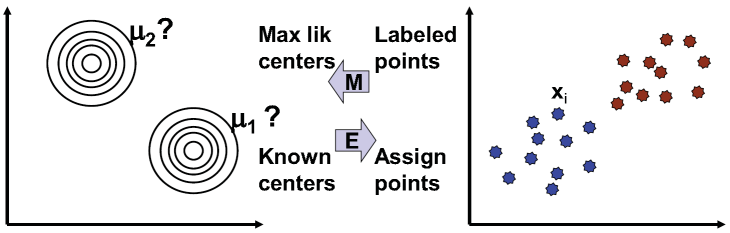


Figure 14.8: K-Means as an expectation maximization (EM) algorithm.

Update rule	Update assignments (E step) ➔ Estimate hidden labels	Algorithm implementing E step in each of the three settings			Update model parameters (M step) ➔ max likelihood
		Expression clustering	HMM learning	Motif discovery	
The hidden label is:		Cluster labels	State path π	Motif positions	
Pick a best	Assign each point to best label	**K-means:** Assign each point to nearest cluster	**Viterbi training:** label sequence with best path	**Greedy:** Find best motif match in each sequence	Average of those points assigned to label
Average all	Assign each point to all labels, probabilistically	**Fuzzy K-means:** Assign to all clusters, weighted by proximity	**Baum-Welch training:** label sequence w all paths (posterior decoding)	**MEME:** Use all positions as a motif occurrence weighed by motif match score	Average of all points, weighted by membership
Sample one	Pick one label at random, based on their relative probability	**N/A:** Assign to a random cluster, sample by proximity	**N/A:** Sample a single label for each position, according to posterior prob.	**Gibbs sampling:** Use one position for the motif, by sampling from the match scores	Average of those points assigned to label(a sample)

Figure 14.9: Comparison of clustering, HMM and motif discovery with respect to expectation minimization (EM) algorithm.

of probabilities.

14.5.4 The limitations of the K-Means algorithm

The k-means algorithm has a few limitations which are important to keep in mind when using it and before choosing it. First of all, it requires a metric. For example, we cannot use the k-means algorithm on a set of words since we would not have any metric.

The second main limitation of the k-means algorithm is its sensitivity to noise. One way to try to reduce the noise is to run a principle component analysis beforehand. Another way is to weight each variable in order to give less weight to the variables affected by significant noise: the weights will be calculated dynamically at each iteration of the algorithm K-means [3].

Third limitation, the choice of initial centers influences the results. There exist heuristics to select the initial cluster centers, but none of them are perfect.

Lastly, we need to guess a priori the number of classes. As we have seen, there are ways to circumvent this problem, essentially by running several times the algorithm while varying k or using the rule of thumb $k \approx \sqrt{n/2}$ if we are short on the computational side. http://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set summarizes well the different techniques to select the number of clusters. Hierarchical clustering provides a handy approach to choosing the number of cluster.

14.5.5 Hierarchical Clustering

While the clustering discussed thus far often provide valuable insight into the nature of various data, they generally overlook an essential component of biological data, namely the idea that similarity might exist on multiple levels. To be more precise, similarity is an intrinsically hierarchical property, and this aspect is not addressed in the clustering algorithms discussed thus far. Hierarchical clustering specifically addresses this in a very simple manner. As illustrated in figure 14.10, it is implemented as follows:

1. *Initialization*: Initialize a list containing each point as an independent cluster.
2. *Iteration*: Create a new cluster containing the two closest clusters in the list. Add this new cluster to the list and remove the two constituent clusters from the list.

Of course, a method for determining distances between clusters is required. The particular metric used varies with context, but some common implementations include the maximum, minimum, and average distances between constituent clusters, and the distance between the centroids of the clusters.

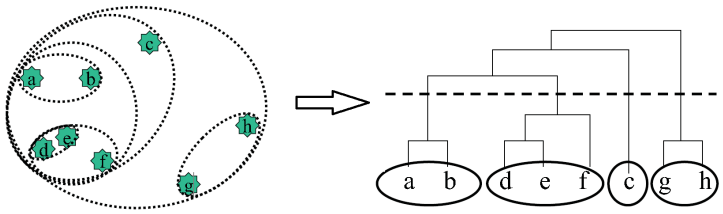


Figure 14.10: Hierarchical Clustering

14.5.6 Evaluating Cluster Performance

The validity of a particular clustering can be evaluated in a number of different ways. The overrepresentation of a known group of genes in a cluster, or, more generally, correlation between the clustering and confirmed biological associations, is a good indicator of validity and significance. If biological data is not yet available, however, there are ways to assess validity using statistics. For instance, robust clusters will appear from clustering even when only subsets of the total available data are used to generate clusters. In addition, the statistical significance of a clustering can be determined by calculating the probability of a particular distribution having been obtained randomly for each cluster. This calculation utilizes variations on the hypergeometric distribution. http://en.wikipedia.org/wiki/Cluster_analysis#Evaluation_of_clustering_results gives several formula to assess the quality of the clustering.

14.6 Current Research Directions

The most significant problems associated with clustering now are associated with scaling existing algorithms cleanly with two attributes: size and dimensionality. To deal with larger and larger datasets, algorithms such as canopy clustering have been developed, in which datasets are coarsely clustered in a manner intended to pre-process the data, following which standard clustering algorithms (e.g. k-means) are applied to subdivide the various clusters. Increase in dimensionality is a much more frustrating problem, and attempt to remedy this usually involve a two stage process in which appropriate relevant subspaces are first identified by appropriate transformations on the original space and then subjected to standard clustering algorithms.

14.7 Further Reading

- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition, February 2009. Found online at http://www-stat.stanford.edu/~tibs/ElemStatLearn/download.html

- Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press.

- McLachlan, G.J. and Basford, K.E. (1988) "Mixture Models: Inference and Applications to Clustering", Marcel Dekker.

- http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

14.8 Resources

- Cluster 3.0: open source clustering software that implements the most commonly used clustering methods for gene expression data analysis.

- MATLAB: K-means clustering: http://www.mathworks.com/help/stats/kmeans.html ; Fuzzy C-means clustering: http://www.mathworks.com/help/fuzzy/fcm.html; Hierarchical Clustering: http://www.mathworks.com/help/stats/linkage.html

- Orange is a free data mining software suite (see module orngClustering for scripting in Python): http://bonsai.hgc.jp/~mdehoon/software/cluster/software.htm

- R (see Cluster Analysis and Finite Mixture Models)

- SAS CLUSTER

14.9 What Have We Learned?

To summarize, in this chapter we have seen that:

- In *clustering*, we identify structure in unlabeled data. For example, we might use clustering to identify groups of genes that display similar expression profiles.
  - Partitioning clustering algorithms, construct non-overlapping clusters such that each item is assigned to exactly one cluster. Example: k-means
  - Agglomerative clustering algorithms construct a hierarchical set of nested clusters, indicating the relatedness between clusters. Example: hierarchical clustering

- In *classification*, we partition data into known labels. For example, we might construct a classifier to partition a set of tumor samples into those likely to respond to a given drug and those unlikely to respond to a given drug based on their gene expression profiles. We will focus on classification in the next chapter.

Bibliography

[1] http://en.wikipedia.org/wiki/File:Heatmap.png.

[2] http://genome.ucsc.edu/ENCODE/.

[3] J.Z. Huang, M.K. Ng, Hongqiang Rong, and Zichen Li. Automated variable weighting in k-means type clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(5):657 –668, may 2005.

[4] Christopher A. Maher, Chandan Kumar-Sinha, Xuhong Cao, Shanker Kalyana-Sundaram, Bo Han, Xiaojun Jing, Lee Sam, Terrence Barrette, Nallasivam Palanisamy, and Arul M. Chinnaiyan. Transcriptome sequencing to detect gene fusions in cancer. *Nature*, 458(7234):97–101, Mar 05 2009.

CHAPTER

FIFTEEN

GENE REGULATION 2 –CLASSIFICATION

Arvind Thiagarajan (athiagar@mit.edu
Fulton Wang (fultonw@mit.edu)
Salil Desai
David Charlton
Kevin Modzelewski
Robert Toscano

15.1 Introduction

In the previous chapter we looked at *clustering*, which provides a tool for analyzing data without any prior knowledge of the underlying structure. As we mentioned before, this is an example of "unsupervised" learning. This chapter deals with supervised learning, in which we are able to use pre-classified data to construct a model by which to classify more datapoints. In this way, we will use existing, known structure to develop rules for identifying and grouping further information.

There are two ways to do classification. The two ways are analogous to the two ways in which we perform motif discovery: HMM, which is a generative model that allows us to actually describe the probability of a particular designation being valid, and CRF, which is a discriminative method that allows us to distinguish between objects in a specific context. There is a dichotomy between generative and discriminative approaches. We will use a Bayesian approach to classify mitochondrial proteins, and SVM to classify tumor samples.

In this lecture we will look at two new algorithms: a generative classifier, Nave Bayes, and a discriminative classifier, Support Vector Machines (SVMs). We will discuss biological applications of each of these models, specifically in the use of Nave Bayes classifiers to predict mitochondrial proteins across the genome and the use of SVMs for the classification of cancer based on gene expression monitoring by DNA microarrays. The salient features of both techniques and caveats of using each technique will also be discussed.

Like with clustering, classification (and more generally supervised learning) arose from efforts in Artificial

Intelligence and Machine Learning. Furthermore, much of the motivating infrastructure for classification had already been developed by probability theorists prior to the advent of either AI or ML.

15.2 Classification - Bayesian Techniques

Consider the problem of identifying mitochondrial proteins. If we look at the human genome, how do we determine which proteins are involved in mitochondrial processes, or more generally which proteins are targeted to the mitochondria1?[1] This is particularly useful because if we know the mitochondrial proteins, we can study how these proteins mediate disease processes and metabolic functions. The classification method we will look considers 7 features for all human proteins:

1. targeting signal
2. protein domains
3. co-expression
4. mass spectrometry
5. sequence homology
6. induction
7. motifs

Our overall approach will be to determine how these features are distributed for both mitochondrial and non-mitochondrial proteins. Then, given a new protein, we can apply probabilistic analysis to these seven features to decide which class it most likely falls into.

15.2.1 Single Features and Bayes Rule

Let's just focus on one feature at first. We must first assume that there is a class dependent distribution for the features. We must first derive this distribution from real data. The second thing we need is the a priori chance of drawing a sample of particular class before looking at the data. The chance of getting a particular class is simply the relative size of the class. Once we have these probabilities, we can use Bayes rule to get the probability a sample is in a particular class given the data(this is called the posterior). We have forward generative probabilities, and use Bayes rules to perform the backwards inference. Note that it is not enough to just consider the probability the feature was drawn from each class dependent distribution, because if we knew a priori that one class(say class A) is much more common than the other, then it should take overwhelming evidence that the feature was drawn from class B's distribution for us to believe the feature was indeed from class B. The correct way to find what we need based on both evidence and prior knowledge is to use Bayes Rule:

$$P(\text{Class}|\text{feature}) = \left(\frac{P(\text{feature}|\text{Class})P(\text{Class})}{P(\text{feature})}\right)$$

[1] Mitochondria is the energy producing machinery of cell. Very early in life, the mitochondria was engulfed by the predecessor to modern day eukaryotes, and now, we have different compartments in our cells. So the mitochonria has its own genome, but it is very depleted from its own ancestral genome - only about 11 genes remain. But there are hundreds are genes that make the mitochondria work, and these proteins are encoded by genes transcribed in the nucleus, and then transported to the mitochondria. So the goal is to figure out which proteins encoded in the genome are targeted to the mitochondria. This is important because there are many diseases associated with the mitochonria, such as aging.

- **Posterior** : $P(\text{Class}|\text{feature})$
- **Prior** : $P(\text{Class})$
- **Likelihood** : $P(\text{feature}|\text{Class})$

This formula gives us exactly the connection we need to flip known feature probabilities into class probabilities for our classifying algorithm. It lets us integrate both the likelihood we derive from our observations and our prior knowledge about how common something is. In the case of mtDNA, for example, we can estimate that mitochondrial DNA makes up something less than 10% of the human genome. Therefore, applying Bayes rule, our classifier should only classify a gene as mitochondrial if there is a very strong likelihood based on the observed features, since the prior probability that any gene is mitochondrial is so low.

With this rule, we can now form a maximum likelihood rule for predicting an objects class based on an observed feature. We want to choose the class that has the highest probability given the observed feature, so we will choose Class1 instead of Class2 if:

$$\left(\frac{P(\text{feature}|\text{Class1})P(\text{Class1})}{P(\text{feature})}\right) > \left(\frac{P(\text{feature}|\text{Class2})P(\text{Class2})}{P(\text{feature})}\right)$$

Notice that $P(\text{feature})$ appears on both sides, so we can cancel that out entirely, and simply choose the class with the highest value of $P(\text{feature}|\text{Class})P(\text{Class})$.

Another way of looking at this is as a discriminant function: By rearranging the formulas above and taking the logarithm, we should select Class1 instead of Class2 precisely when

$$\log\left(\frac{P(\text{X}|\text{Class1})P(\text{Class1})}{P(\text{X}|\text{Class2})P(\text{Class2})}\right) > 0$$

In this case the use of logarithms provide distinct advantages:

1. Numerical stability
2. Easier math (its easier to add the expanded terms than multiply them)
3. Monotonically increasing discriminators.

This discriminant function does not capture the penalties associated with misclassification (in other words, is one classification more detrimental than other). In this case, we are essentially minimizing the number of misclassifications we make overall, but not assigning penalties to individual misclassifications. From examples discussed in class and in the problem set - if we are trying to classify a patient as having cancer or not, it could be argued that it is far more harmful to misclassify a patient as being healthy if they have cancer than to misclassify a patient as having cancer if they are healthy. In the first case, the patient will not be treated and would be more likely to die, whereas the second mistake involves emotional grief but no greater chance of loss of life. To formalize the penalty of misclassification we define something called a loss function,L_{kf} , which assigns a loss to the misclassification of an object as class j when the true class is class k (a specific example of a loss function was seen in Problem Set 2).

15.2.2 Collecting Data

The preceding tells us how to handle predictions if we already know the exact probabilities corresponding to each class. If we want to classify mitochondrial proteins based on feature X, we still need ways of determining

the probabilities $P(\text{mito})$, $P(\text{ mito})$, $P(X|\text{mito})$ and $P(X|\text{ mito})$. To do this, we need a training set: a set of data that is already classified that our algorithm can use to learn the distributions corresponding to each class. A high-quality training set (one that is both large and unbiased) is the most important part of any classifier. An important question at this point is, how much data do we need about known genes in order to build a good classifier for unknown genes? This is a hard question whose answer is not fully known. However, there are some simple methods that can give us a good estimate: when we have a fixed set of training data, we can keep a holdout set that we dont use for our algorithm, and instead use those (known) data points to test the accuracy of our algorithm when we try to classify them. By trying different sizes of training versus *holdout* set, we can check the accuracy curve of our algorithm. Generally speaking, we have enough training data when we see the accuracy curve flatten out as we increase the amount of training data (this indicates that additional data is likely to give only a slight marginal improvement). The holdout set is also called the test set, because it allows us to test the generalization power of our classifier.

Supposing we have already collected our training data, however, how should we model $P(X|Class)$? There are many possibilities. One is to use the same approach we did with clustering in the last lecture and model the feature as a Gaussian then we can follow the maximum likelihood principle to find the best center and variance. The one used in the mitochondrial study is a simple density estimate: for each feature, divide the range of possibilities into a set of bins (say, five bins per feature). Then we use the given data to estimate the probability of a feature falling into each bin for a given class. The principle behind this is again maximum likelihood, but for a multinomial distribution rather than a Gaussian. We may choose to discretize a otherwise continuous distribution because estimating a continuous distribution can be complex.

There is one issue with this strategy: what if one of the bins has zero samples in it? A probability of zero will override everything else in our formulas, so that instead of thinking this bin is merely unlikely, our classifier will believe it is *impossible*. There are many possible solutions, but the one taken here is to apply the *Laplace Correction*: add some small amount (say, one element) into each bin, to draw probability estimates slightly towards uniform and account for the fact that (in most cases) none of the bins are truly impossible. Another way to avoid having to apply the correction is to choose bins that are not too small so that bins will not have zero samples in them in practice. If you have many many points, you can have more bins, but run the risk of overfitting your training data.

15.2.3 Estimating Priors

We now have a method for approximating the feature distribution for a given class, but we still need to know the relative probability of the classes themselves. There are three general approaches:

1. Estimate the priors by counting the relative frequency of each class in the training data. This is prone to bias, however, since data available is often skewed disproportionately towards less common classes (since those are often targeted for special study). If we have a high-quality (representative) sample for our training data, however, this works very well.

2. Estimate from expert knowledge—there may be previous estimates obtained by other methods independent of our training data, which we can then use as a first approximation in our own predictions. In other words, you might ask experts what the percentage of mitochondrial proteins are.

3. Assume all classes are equally likely we would typically do this if we have no information at all about the true frequencies. This is effectively what we do when we use the maximum likelihood principle: our clustering algorithm was essentially using Bayesian analysis under the assumption that all priors are equal. This is actually a strong assumption, but when you have no other data, this is the best you can do.

For classifying mitochondrial DNA, we use method (2), since some estimates on the proportions of mtDNA were already known. But there is an complication there are more than 1 features.

15.2.4 Multiple features and Nave Bayes

In classifying mitochondrial DNA, we were looking at 7 features and not just one. In order to use the preceding methods with multiple features, we would need not just one bin for each individual feature range, but one for each combination of features if we look at two features with five ranges each, thats already 25 bins. All seven features gives us almost 80,000 bins and we can expect that most of those bins will be empty simply because we dont have enough training data to fill them all. This would cause problems because zeroes cause infinite changes in the probabilities of being in one class. Clearly this approach wont scale well as we add more features, so we need to estimate combined probabilities in a better way.

The solution we will use is to assume the features are independent, that is, that once we know the class, the probability distribution of any feature is unaffected by the values of the other features. This is the Nave Bayes Assumption, and it is almost always false, but it is often used anyway for the combined reasons that it is very easy to manipulate mathematically and it is often close enough to the truth that it gives a reasonable approximation. (Note that this assumption does not say that all features are independent: if we look at the overall model, there can be strong connections between different features, but the assumption says that those connections are divided by the different classes, and that within each individual class there are no further dependencies.) Also, if you know that some features are coupled, you could learn the joint distribution in only some pairs of the features.

Once we assume independence, the probability of combined features is simply the product of the individual probabilities associated with each feature. So we now have:

$$P(f_1, f_2, K, f_N|\text{Class}) = P(f_1|\text{Class})P(f_2|\text{Class})KP(f_N|\text{Class})$$

Where f_1 represents feature 1. Similarly, the discriminant function can be changed to the multiplication of the prior probabilities:

$$G(f_1, f_2, K, f_N) = \log\left(\frac{\Pi P(f_1|\text{Class1})P(\text{Class1})}{\Pi P(f_1|\text{Class2})P(\text{Class2})}\right)$$

15.2.5 Testing a classifier

A classifier should always be tested on data not contained in its training set. We can imagine in the worst case an algorithm that just memorized its training data and behaved randomly on anything else a classifier that did this would perform perfectly on its training data, but that indicates nothing about its real performance on new inputs. This is why its important to use a test, or holdout, set as mentioned earlier. However, a simple error rate doesnt encapsulate all of the possible consequences of an error. For a simple binary classifier (an object is either in or not in a single target class), there are the following for types of errors:

1. True positive (TP)
2. True negative (TN)
3. False positive (FP)
4. False negative (FN)

The frequency of these errors can be encapsulated in performance metrics of a classifier which are defined as,

1. *Sensitivity* what fraction of objects that are in a class are correctly labeled as that class? That is, what fraction have true positive results? High sensitivity means that elements of a class are very likely to be labeled as that class. Low sensitivity means there are too many false negatives.
2. *Specificity* what fraction of objects not in a class are correctly labeled as not being in that class? That is, what fraction have true negative results? High specificity means that elements labeled as belonging to a class are very likely to actually belong to it. Low specificity means there are too many false positives.

In most algorithms there is a tradeoff between sensitivity and specificity. For example, we can reach a sensitivity of 100% by labeling everything as belonging to the target class, but we will have a specificity of 0%, so this is not useful. Generally, most algorithms have some probability cutoff they use to decide whether to label an object as belonging to a class (for example, our discriminant function above). Raising that threshold increases the specificity but decreases the sensitivity, and decreasing the threshold does the reverse. The MAESTRO algorithm for classifying mitochondrial proteins (described in this lecture) achieves 99% specificity and 71% sensitivity.

15.2.6 MAESTRO Mitochondrial Protein Classification

They find a class dependent distribution for each features by creating several bins and evaluating the proportion of mitochondrial and non mitochondrial proteins in each bin. This lets you evaluate the usefulness of each feature in classification. You end up with a bunch of medium strength classifiers, but when you combine them together, you hopefully end up with a stronger classifier. Calvo et al. [1] sought to construct high-quality predictions of human proteins localized to the mitochondrion by generating and integrating data sets that provide complementary clues about mitochondrial localization. Specifically, for each human gene product p, they assign a score $s_i(p)$, using each of the following seven genome-scale data sets targeting signal score, protein domain score, cis-motif score, yeast homology score, ancestry score, coexpression score, and induction score (details of each of the meaning and content of each of these data sets can be found in the manuscript). Each of these scores $s_1 - S_7$ can be used individually as a weak genome-wide predictor of mitochondrial localization. Each methods performance was assessed using large gold standard curated training sets - 654 mitochondrial proteins T_{mito} maintained by the MitoP2 database1 and 2,847 nonmitochondrial proteins T_{mito} annotated to localize to other cellular compartments. To improve prediction accuracy, the authors integrated these eight approaches using a nave Bayes classifier that was implemented as a program called MAESTRO. So we can take several weak classifiers, and combine them to get a stronger classifier.

When MAESTRO was applied across the human proteome, 1451 proteins were predicted as mitochondrial proteins and 450 novel proteins predictions were made. As mentioned in the previous section The MAESTRO algorithm achieves a 99% specificity and a 71% sensitivity for the classification of mitochondrial proteins, suggesting that even with the assumption of feature independence, Nave Bayes classification techniques can prove extremely powerful for large-scale (i.e. genome-wide) scale classification.

15.3 Classification Support Vector Machines

The previous section looked at using probabilistic (or generative) models for classification, this section looks at using discriminative techniques in essence, can we run our data through a function to determine its

structure? Such discriminative techniques avoid the inherent cost involved in generative models which might require more information than is actually necessary.

Support vector machine techniques essentially involve drawing a vector thats perpendicular to the line(hyperplane) separating the training data. The approach is that we look at the training data to obtain a separating hyperplane so that two classes of data lie on different sides of the hyperplane. There are, in general, many hyperplanes that can separate the data, so we want to draw the hyperplane that separates the data the most - we wish to choose the line that maximizes the distance from the hyperplane to any data point. In other words, the SVM is a maximum margin classifier. You can think of the hyperplane being surrounded with margins of equal size on each side of the line, with no data points inside the margin on either side. We want to draw the line that allows us to draw the largest margin. Note that once the separating line and margin are determined, some data points will be right on the boundary of the margin. These are the data points that keep us from expanding the margin any further, and thus determine the line/margin. Such points are called the support vectors. If we add new data points outside the margin or remove points that are not support vectors, we will not change the maximum margin we can achieve with any hyperplane.

Suppose that the vector perpendicular to the hyperplane is w, and that the hyperplane passes through the point $\left(\frac{b}{|w|}\right)$. Then a point x is classified as being in the positive class if $w * x$ is greater than b, and negative otherwise. It can be shown that the optimal w, that is, the hyperplane that achieves the maximum margin, can actually be written as a linear combination of the data vectors $\Sigma a_i * x_i$. Then, to classify a new data point x, we need to take the dot product of w with x to arrive at a scalar. Notice that this scalar, $\Sigma a_i * (x_i * x)$ only depends on the dot product between x and the training vectors x_is. Furthermore, it can be shown that finding the maximum margin hyperplane for a set of (training) points amounts to maximizing a linear program where the objective function only depends on the dot product of the training points with each other. This is good because it tells us that the complexity of solving that linear program is independent of the of dimension of the data points. If we precompute the pairwise dot products of the training vectors, then it makes no difference what the dimensionality of the data is in regards to the running time of solving the linear program.

15.3.1 Kernels

We see that SVMs are dependent only on the dot product of the vectors. So, if we call our transformation $\phi(v)$, for two vectors we only care about the value of $\phi(v_1) \cdot \phi(v_2)$ The trick to using kernels is to realize that for certain transformations ϕ, there exists a function $K(v_1, v_2)$, such that:

$$K(v_1, v_2) = \phi(v_1) \cdot \phi(v_2)$$

In the above relation, the right-hand side is the dot product of vectors with very high dimension, but the left-hand side is the function of two vectors with lower dimension. In our previous example of mapping $x \to (x, y = x^2)$, we get

$$K(x_1, x_2) = (x_1 x_1^2) \cdot (x_2, x_2^2) = x_1 x_2 + (x_1 x_2)^2$$

Now we did not actually apply the transformation ϕ, we can do all our calculations in the lower dimensional space, but get all the power of using a higher dimension.

Example kernels are the following:

1. Linear kernel: $K(v_1, v_2) = v_1 \cdot v_2$ which represents the trivial mapping of $\phi(x) = x$

2. Polynomial kernel: $K(v_1, v_2) = (1 + v_1 \cdot v_2)^n$ which was used in the previous example with $n = 2$.

3. Radial basis kernel: $K(v_1, v_2) = \exp(-\beta|v_1 - v_2|^2)$ This transformation is actually from a point v_1 to a function (which can be thought of as being a point in Hilbert space) in an infinite-dimensional space. So what were actually doing is transforming our training set into functions, and combining the to get a decision boundary. The functions are Gaussians centered at the input points.

4. Sigmoid kernel: $K(v_1, v_2) = \tanh[\beta(v_1^T v_2 + r)]$ Sigmoid kernels have been popular for use in SVMs due to their origin in neural networks (e.g. sigmoid kernel functions are equivalent to two-level, perceptron neural networks). It has been pointed out in previous work (Vapnik 1995) that the kernel matrix may not be positive semi-definite for certain values of the parameters μ and r. The sigmoid kernel has nevertheless been used in practical applications [2].

Here is a specific example of a kernel function. Consider the two classes of one-dimensional data:

$\{-5, -4, -3, 3, 4, 5\} and \{-2, -1, 0, 1, 2\}$

This data is clearly not linearly separable, and the best separation boundary we can find might be $x > -2.5$. Now consider applying the transformation . The data can now be written as new pairs,

$\{-5, -4, -3, 3, 4, 5\} \rightarrow \{(-5, 25), (-4, 16), (-3, 9), (3, 9), (4, 16), (5, 25)\}$

and

$\{-2, -1, 0, 1, 2\} \rightarrow \{(-2, -4), (-1, 1), (0, 0), (1, 1), (2, 4)\}$

This data is separable by the rule $y > 6.5$, and in general the more dimensions we transform data to the more separable it becomes.

An alternate way of thinking of this problem is to transform the classifier back in to the original low-dimensional space. In this particular example, we would get the rule $x^2 < 6.5$, which would bisect the number line at two points. In general, the higher dimensionality of the space that we transform to, the more complicated a classifier we get when we transform back to the original space.

One of the caveats of transforming the input data using a kernel is the risk of overfitting (or over-classifying) the data. More generally, the SVM may generate so many feature vector dimensions that it does not generalize well to other data. To avoid overfitting, cross-validation is typically used to evaluate the fitting provided by each parameter set tried during the grid or pattern search process. In the radial-basis kernel, you can essentially increase the value of β until each point is within its own classification region (thereby defeating the classification process altogether). SVMs generally avoid this problem of over-fitting due to the fact that they maximize margins between data points.

When using difficult-to-separate training sets, SVMs can incorporate a cost parameter C, to allow some flexibility in separating the categories. This parameter controls the trade-off between allowing training errors and forcing rigid margins. It can thereby create a *soft* margin that permits some misclassifications. Increasing the value of C increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well.

Can we use just any function as our kernel? The answer to this is provided by Mercers Condition which provides us an analytical criterion for choosing an acceptable kernel. Mercers Condition states that a kernel $K(x, y)$ is a valid kernel if and only if the following holds For any $g(x)$ such that $\int g(x)^2 dx$ is finite, we have:

$\int \int K(x,y)g(x)g(y)dxdy \geq 0$[3]

In all, we have defined SVM discriminators and shown how to perform classification with appropriate kernel mapping functions that allow performing computations on lower dimension while being to capture all the information available at higher dimensions. The next section describes the application of SVMs to the classification of tumors for cancer diagnostics.

15.4 Tumor Classification with SVMs

A generic approach for classifying two types of acute leukemias acute myeloid leukemia (AML) and acute lymphoid leukemia (ALL) was presented by Golub et al. [4]. This approach centered on effectively addressing three main issues:

1. Whether there were genes whose expression pattern to be predicted was strongly correlated with the class distinction (i.e. can ALL and AML be distinguished)
2. How to use a collection of known samples to create a "class predictor" capable of assigning a new sample to one of two classes
3. How to test the validity of their class predictors

They addressed (1) by using a "neighbourhood analysis" technique to establish whether the observed correlations were stronger than would be expected by chance. This analysis showed that roughly 1100 genes were more highly correlated with the AML-ALL class distinction than would be expected by chance. To address (2) they developed a procedure that uses a fixed subset of "informative genes" (chosen based on their correlation with the class distinction of AML and ALL) and makes a prediction based on the expression level of these genes in a new sample. Each informative gene casts a "weighted vote" for one of the classes, with the weight of each vote dependent on the expression level in the new sample and the degree of that genes correlation with the class distinction. The votes are summed to determine the winning class. To address (3) and effectively test their predictor by first testing by cross-validation on the initial data set and then assessing its accuracy on an independent set of samples. Based on their tests, they were able to identify 36 of the 38 samples (which were part of their training set!) and all 36 predictions were clinically correct. On the independent test set 29 of 34 samples were strongly predicted with 100% accuracy and 5 were not predicted.

A SVM approach to this same classification problem was implemented by Mukherjee et al.[5]. The output of classical SVM is a binary class designation. In this particular application it is particularly important to be able to reject points for which the classifier is not confident enough. Therefore, the authors introduced a confidence interval on the output of the SVM that allows for rejection of points with low confidence values. As in the case of Golub et al.[4] it was important for the authors to infer which genes are important for the classification. The SVM was trained on the 38 samples in the training set and tested on the 34 samples in the independent test set (exactly in the case of Golub et al.). The authors results are summarized in the following table (where $|d|$ corresponds to the cutoff for rejection).

Genes	**Rejects**	**Errors**	**Confidence level**	$\|d\|$
7129	3	0	93%	0.1
40	0	0	93%	0.1
5	3	0	92%	0.1

These results a significant improvement over previously reported techniques, suggesting that SVMs play an important role in classification of large data sets (as those generated by DNA microarray experiments).

15.5 Semi-Supervised Learning

In some scenarios we have a data set with only a few labeled data points, a large number of unlabeled data points and inherent structure in the data. This type of scenario both clustering and classification do not perform well and a hybrid approach is required. This semi-supervised approach could involve the clustering of data first followed by the classification of the generated clusters.

15.5.1 Open Problems

15.6 Current Research Directions

15.7 Further Reading

- Richard O. Duda, Peter E. Hart, David G. Stork (2001) Pattern classification (2nd edition), Wiley, New York
- See previous chapter for more books and articles.

15.8 Resources

- Statistical Pattern Recognition Toolbox for Matlab.
- See previous chapter for more tools

Bibliography

[1] Calvo, S., Jain, M., Xie, X., Sheth, S.A., Chang, B., Goldberger, O.A., Spinaz- zola, A., Zeviani, M., Carr, S.A., and Mootha, V.K. (2006). Systematic identifi- cation of human mitochondrial disease genes through integrative genomics. Nat. Genet. 38, 576582.

[2] Scholokopf, B., et al., 1997. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Transactions on Signal Processing.

[3] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[4] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, and C. D. Bloomfield. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[5] S. Mukherjee, P. Tamayo, D. Slonim, A. Verri, T. Golub, J. P. Mesirov, and T. Poggio. Support vector machine classification of microarray data. Technical report, AI Memo 1677, Massachusetts Institute of Technology, 1998.

CHAPTER

SIXTEEN

REGULATORY MOTIFS, GIBBS SAMPLING, AND EM

James Yeung (2012)
Yinqing Li and Arvind Thiagarajan (2011)
Bianca Dumitrascu and Neal Wadhwa (2010)
Joseph Lane (2009)
Brad Cater and Ethan Heilman (2008)

Figures

16.1 Introduction to regulatory motifs and gene regulation

16.1.1 SLIDE: The regulatory code: All about regulatory motifs

We have already explored the areas of dynamic programming, sequence alignment, sequence classification and modeling, hidden Markov models, and expectation maximization. As we will see, these techniques will be useful in identifying novel motifs and elucidating their functions.

Motifs are short (6-8 bases long), recurring patterns that have well-defined biological functions. Motifs include DNA patterns in enhancer regions, promoter motifs, and motifs on the RNA level such as splicing signals. They can, for instance, indicate sequence-specific binding sites for proteins such as nucleases and transcription factors. As we have discussed, genetic activity is regulated in response to environmental variations. Transcription Factors, the proteins that accomplish this regulation, are recruited to their target genes by different varieties of regulatory motifs. In general, the transcription factors bind to specific motifs. Once bound, the transcription factors activate or repress the expression of the associated gene.

16.1.2 Two settings: co-regulated genes (EM,Gibbs), de novo

Transcription factors often regulate a group of genes that are involved in similar cellular processes. Thus, genes that contain the same motif in their upstream regions are likely to be related in their functions. In fact, many regulatory motifs are identified by analyzing the upstream regions of genes known to have similar functions.

Motifs have become exceedingly useful for defining genetic regulatory networks and deciphering the functions of individual genes. With our current computational abilities, regulatory motif discovery and analysis has progressed considerably and remains at the forefront of genomic studies.

Transcription factors (TFs) are proteins that bind to specific regulatory motifs, in particular motifs responsible for controlling gene expression. Specific TFs are in some sense complementary to the motifs which they recognize, and so they are able to locate and bind to their motifs without disrupting DNA structure in the process. Depending on the structural configuration of the motif and it's interaction with the corresponding TF, the activity of the controlled gene can either be elevated or depressed. TFs use several mechanisms in order to control gene expression, including acetylation and deacetylation of histone proteins, recruitment of cofactor molecules to the TF-DNA complex, and stabilization or disruption of RNA-DNA interfaces during transcription.

Motifs are also recognized by microRNAs, which bind to motifs given strict conditions of complementarity, Nucleosomes, which recognize motifs based on their GC content, and even RNAs.

Before we can get into algorithms for motif discovery, we must first understand the characteristics of motifs, especially those that make motifs somewhat difficult to find. As mentioned above, motifs are generally very short, usually only 6-8 base pairs long. Additionally, motifs can be degenerate, i.e. only the nucleotides at certain locations within the motif affect the motif's function. This degeneracy arises because transcription factors are free to interact with their corresponding motifs in manners more complex than a simple complementarity relation. As seen in 16.1, a protein may only interact with a portion of the motif (sides or center), rendering the rest of the motif redundant. The physical structure of the protein determines which parts of the motif are critical for the recognition and subsequent interaction. For instance, a protein may be sensitive to the presence of a purine as opposed to a pyrimidine, while unable to distinguish between different purines and different pyrimidines. The topology of the transcription factors also affects

the interaction between motif and factor.

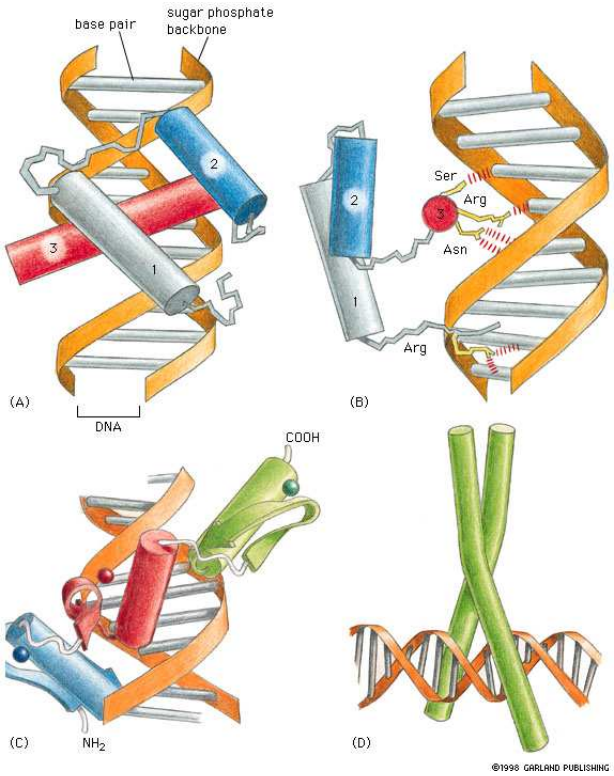


Figure 16.1: Transcription factors binding to DNA at a motif site

The issue of degeneracy within a motif poses a challenging problem. If we were only looking for a fixed k-mer, we could simply search for the k-mer in all the sequences we are looking at using local alignment tools. However, the motif may vary from sequence to sequence. Because of this, a string of nucleotides that is known to be a regulatory motif is said to be an *instance* of a motif because it represents one of potentially many different combinations of nucleotides that fulfill the function of the motif.

In our approaches, we make two assumptions about the data. First, we assume that there are no pairwise correlations between bases, i.e. that each base is independent of every other base. While such correlations do exist in real life, considering them in our analysis would lead to an exponential growth of the parameter space being considered, and consequently we would run the risk of overfitting our data. The second assumption we make is that all motifs have fixed lengths; indeed, this approximation simplifies the problem greatly. Even with these two assumptions, however, motif finding is still a very challenging problem. The relatively small size of motifs, along with their great variety, makes it faily difficult to locate them. In addition, a motif's location relative to the corresponding gene is far from fixed; the motif can be upstream or downstream, and the distance between the gene and the motif also varies. Indeed, sometimes the motif is up to $10k$ to $10M$ base pairs from the gene.

16.1.3 SLIDE: Starting positions / Motif matrix

Because motif instances exhibit great variety, we generally use a profile matrix to characterize the motif. This matrix gives the frequency of each base at each location in the motif. In the adjacent figure, there is an example profile matrix. We denote the profile matrix by p_{ck} where k corresponds to the position within the motif, with p_{c0} denoting the distribution of bases in non-motif regions.

sequence positions

	1	2	3	4	5	6	7	8
A	0.1	0.3	0.1	0.2	0.2	0.4	0.3	0.1
C	0.5	0.2	0.1	0.1	0.6	0.1	0.2	0.7
G	0.2	0.2	0.6	0.5	0.1	0.2	0.2	0.1
T	0.2	0.3	0.2	0.2	0.1	0.3	0.3	0.1

Figure 16.2: Example Profile Matrix

We now define the problem of motif finding more rigorously. We assume that we are given a set of co-regulated and functionally related genes. In the past, many motifs were discovered by doing footprint experiments, in which parts of sequence to which transcription factors were binding would be isolated. Since motifs are highly involved in the regulation of protein building, these regions were likely to correspond to motifs. There are several computational methods that can be used to locate motifs:

1. Perform a local alignment across the set of sequences and explore the alignments that resulted in a very high alignment score.

2. Model the promoter regions using an Hidden Markov Model and then use a generative model to find non-random sequences.

3. Reduce the search space by applying expert knowledge for what motifs should look like.

4. Search for conserved blocks from two different sequences.

5. Count the number of words that occur in each region that is highly likely to contain a motif.

6. Use probabilistic methods, such as EM, Gibbs Sampling, or a greedy algorithm

Before utilizing the fifth option listed, there are a few difficulties that must be considered. For example, there could be many common words that occur in these regions that are in fact not regulatory motifs but instead different sets of instructions. Furthermore, given a list of words that could be a motif, it is unclear whether the correct selection of motif is the most common word or the least common word. While motifs are generally overrepresented in promoter regions, transcription factors may be unable to bind if an excess of motifs are present. One possible resolution to this choice might be to find the sequence with maximum relative frequency in promoter regions over other regions. This strategy is commonly performed as a post processing step to narrow down the number of possible motifs.

In fact, the strategy has its biological relevance. In 2003, Professor Kellis argued that there must be some selective pressure to cause a particular sequence to be occur on specific places. His PhD. thesis on the topic can be found at the following location: http://web.mit.edu/manoli/www/publications/Kellis_Thesis_03.pdf.

In the next section, we will talk more about EM, Gibbs Sampling, and the Greedy Algorithm. Thse algorithms are the focus of this lecture.

16.2 Expectation maximization: Motif matrix and positions

We are given a set of sequences with the assumption that motifs are enriched in them. The task is to find common motif in those sequences. In EM, Gibbs Sampling and the Greedy Algorithm, we first compute a profile matrix and then use this to compute the matrix Z_{ij}, which is the probability that there is a motif instance at position j in the sequence i. This Z matrix is then used to recompute the original profile matrix until convergence. Some examples of this matrix are graphically represented by Figure 16.3

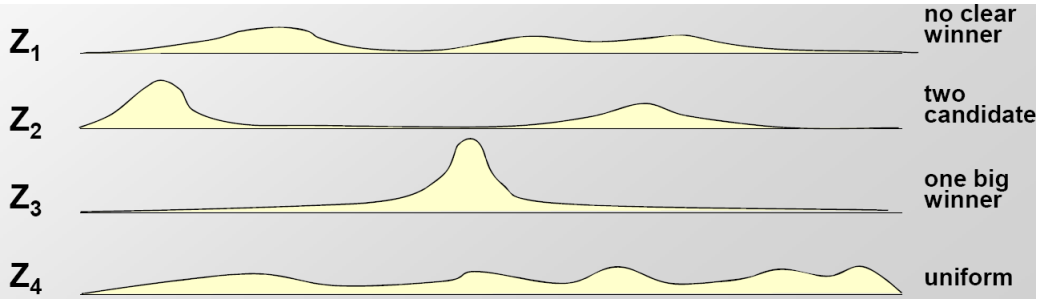


Figure 16.3: Examples of the Z matrix computed

.

16.2.1 SLIDE: Representing the starting position probabilities (Z_{ij})

Intuitvely, the greedy algorithm will always pick the most probable location for the motif. The EM algorithm will take an average while Gibbs Sampling will actually use the probability distribution given by Z to sample a motif in a step.

16.2.2 E step: Estimate motif positions Z_{ij} from motif matrix

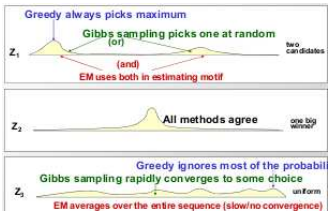


Figure 16.4: Selecting motif location: the greedy algorithm will always pick the most probable location for the motif. The EM algorithm will take an average while Gibbs Sampling will actually use the probability distribution given by Z to sample a motif in each step

Step 1: Initialization The first step in EM is to generate an initial probability weight matrix (PWM). The PWM describes the frequency of each nucleotide at each location in the motif. In 16.5, there is an example of a PWM. In this example, we assume that the motif is eight bases long.

If you are given a set of aligned sequences and the location of suspected motifs within them, then finidng the PWM is accomplished by counting the number of bases in each position in every suspected motif. We can initialize PWM by choosing starting locations randomly. It is important to realize that the result of the EM is entirely dependent on the initial parameters, that is the initial value of the

PWM. It is good practice to run the EM algorithm multiple times using different initial values. This helps ensure that the solution we find is not only locally optimal, but also globally optimal.

We refer to the PWM as p_{ck}, where p_{ck} is the probability of character c occurring in position k of the motif. Note: if there is 0 probability, it is generally a good idea to insert pseudo-counts into your probabilities. The PWM is also called the profile matrix. In addition to the PWM, we also keep a background distribution $p_{ck,k=0}$, a distribution of the bases not in the motif.

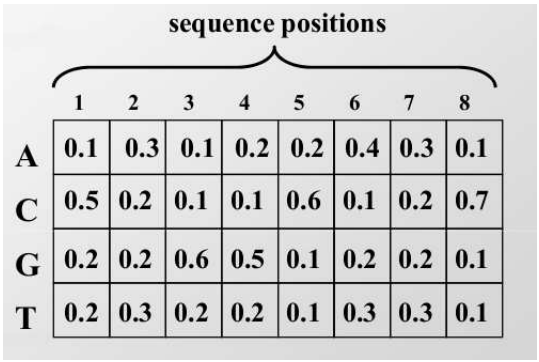


Figure 16.5: Sample position weight matrix

Step 2: Expectation In the expectation step, we generate a vector Z_{ij} which contains the probability of the motif starting in position j in sequence i. In EM, the Z vector gives us a way of classifying all of the nucleotides in the sequences and tell us whether they are part of the motif or not. We can calculate Z_{ij} using Bayes' Rule. This simplifies to:

$Z_{ij}^t = \frac{Pr^t(X_i|Z_{ij})Pr^t(Z_{ij}=1)}{\Sigma_{k=1}^{L-W+1} Pr^t(X_i|Z_{ij}=1)Pr^t(Z_{ik}=1)}$

where $Pr^t(X_i|Z_{ij}=1) = Pr(X_i|Z_{ij}=1,p)$ is defined as

$$\Pr(X_i \mid Z_{ij}=1,p) = \underbrace{\prod_{k=1}^{j-1} p_{c_k,0}}_{\text{before motif}} \underbrace{\prod_{k=j}^{j+W-1} p_{c_k,k-j+1}}_{\text{motif}} \underbrace{\prod_{k=j+W}^{L} p_{c_k,0}}_{\text{after motif}}$$

This probability is the probability of sequence i existing given that the motif is in the position j in the sequence. The first and last product correspond to the probability that most of the sequence comes from some background probability distribution while the middle product corresponds to the motif coming from a motif probability distribution. In this equation, we assume that the sequence has length L and the motif has length W.

16.2.3 M step: Find max-likelihood motif from all positions Zij

Step 3: Maximization Once we have calculated Z^t, we can use the results to update both the PWM and the background probability distribution. We can update the PWM using the following equation

Step 4: Repeat Repeat steps 2 and 3 until convergence.

One possible way to test whether the profile matrix has converged is to measure how much each element in the PWM changes after step maximization. If the change is below a chosen threshold, then we can terminate the algorithm. It is advisable to rerun the algorithm with different intial starting positions to try reduce the chance of converging on a local maximum that is not the global maximum and to get a good sense of the solutions space.

$$p_{c,k}^{(t+1)} = \frac{n_{c,k} + d_{c,k}}{\sum_b (n_{b,k} + d_{b,k})}$$

pseudo-counts

$$n_{c,k} = \begin{cases} \sum_i \sum_{\{j | X_{i,j+k-1}=c\}} Z_{ij} & k > 0 \text{ motif} \\ n_c - \sum_{j=1}^{W} n_{c,j} & k = 0 \end{cases}$$

total # of c's in data set

16.3 Gibbs Sampling: Sample from joint (M,Zij) distribution

16.3.1 Sampling motif positions based on the Z vector

Gibbs sampling is similar to EM except that it is a stochastic process, while EM is deterministic. In the expectation step, we only consider nucleotides within the motif window in Gibbs sampling. In the maximization step, we sample from Z_{ij} and use the result to update the PWM instead of averaging over all values as in EM.

Step 1: Initialization As with EM, you generate your initial PWM with a random sampling of initial starting positions. Unlike EM, the algorithms outcome is not sensitive to the initial PWM. This is because Gibbs sampling updates its PWM based on a random sample. An advantage of this is that the algorithm is more robust to getting stuck in a local maximum.

Step 2: Remove Remove one sequence, X_i, from your set of sequences. You will change the starting location of for this particular sequence.

Step 3: Update Using the remaining set of sequences, update the PWM by counting how often each base occurs in each position.

Step 4: Sample Using the newly updated PWM, generated a probability distribution for starting positions in sequence X_i. To generate each element, A_{ij}, of the probability distribution, the following formula is used,

$$A_{ij} = \frac{\prod_{k=j}^{j+W-1} p_{ck}, k-j+1}{\prod_{k=j}^{j+W-1} p_{ck}, 0}$$

This is simply the probability that the sequence was generated using the motif PWM divided by the probability that the sequence was generated using the background PWM.

Select a new starting position for X_i by randomly sampling from this A_{ij}.

Step 5: Iterate Loop back to Step 2 and iterate the algorithm until convergence.

16.3.2 More likely to find global maximum, easy to implement

Two popular implementations of Gibbs Sampling applied to this problem are AlignACE and BioProspector. A more general Gibbs Sampler can be found in the program WinBUGS. Both AlignACE and BioProspector use the aforementioned algorithm for several choices of initial values and then report common motifs. Gibbs sampling is easier to implement than E-M and in theory, it converges quickly, and it is less likely to get stuck at a local optimum. However, the search is less sytematic.

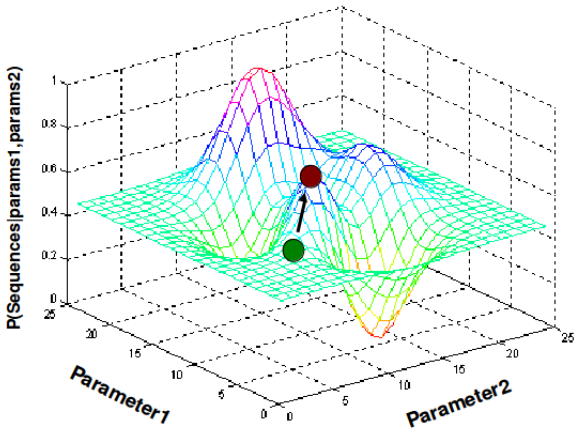


Figure 16.6: Gibbs Sampling

16.4 Evolutionary signatures for de novo motif discovery

As discussed in beginning of this chapter, the core problem for motif finding is to define the criteria that what accounts for a valid motif and where they hide. Computationally, this equals to the selection of input sequences and setting a prior for motif. Since most motifs are linked to important biological functions, as expected, they are usually conserved aross spieces and are called evolutionary signatures. As such, searching motifs through aligned sequences taken from close spieces would increase the rate of finding functional motifs.

16.4.1 Genome-wide conservation scores, motif extension

16.4.2 Validation of discovered motifs: functional datasets

16.5 Evolutionary signatures for instance identification

16.6 Phylogenies, Branch length score Confidence score

16.6.1 Foreground vs. background. Real vs. control motifs.

16.7 Other figures that might be nice:

16.7.1 SLIDE: One solution: search from many different starts

16.7.2 SLIDE: Gibbs Sampling and Climbing

16.7.3 SLIDE: Conservation islands overlap known motifs

16.7.4 SLIDES: Tests 1,2,3 on page 9

16.7.5 SLIDE: Motifs have functional enrichments

16.7.6 SLIDE: Experimental instance identification: ChIP-Chip / ChIP-Seq

16.7.7 SLIDE: Increased sensitivity using BLS

16.8 Possibly deprecated stuff below:

16.8.1 Greedy

While the greedy algorithm is not used very much in practice, it is important know how it functions and mainly its advantages and disadvantages compared to EM and Gibbs sampling. The Greedy algorithm works just like Gibbs sampling except for a main difference in Step 4. Instead of randomly choosing selecting a new starting location, it simply picks the starting location with the highest probability.

This makes the Greedy algorithm quicker, although only slightly, than Gibbs sampling but reduces its chances of finding a global maximum considerably. In cases where the starting location probability distribution is fairly level, the greedy algorithm ignores the weights of every other starting position other

than the most likely.

16.9 OOPS,ZOOPS,TCM

The different types of sequence model make differing assumptions about how and where motif occurrences appear in the dataset. The simplest model type is OOPS (One-Occurence-Per-Sequence) since it assumes that there is exactly one occurrence per sequence of the motif in the dataset. This is the case we have analyzed in the Gibbs sampling section. This type of model was introduced by Lawrence & Reilly (1990) [2], when they describe for the first time a generalization of OOPS, called ZOOPS (Zero-or-One-Occurrence-Per-Sequence), which assumes zero or one motif occurrences per dataset sequence. Finally, TCM (Two-Component Mixture) models assume that there are zero or more non-overlapping occurrences of the motif in each sequence in the dataset, as described by Baily & Elkan (1994). [1] Each of these types of sequence model consists of two components, which model, respectively, the motif and non-motif (background) positions in sequences. A motif is modelled by a sequence of discrete random variables whose parameters give the probabilities of each of the different letters (4 in the case of DNA, 20 in the case of proteins) occurring in each of the different positions in an occurrence of the motif. The background positions in the sequence are modelled by a single discrete random variable.

16.10 Extension of the EM Approach

16.10.1 ZOOPS Model

The approach presented before (OOPS) relies on the assumption that every sequence is characterized by only one motif (e.g., there is exactly one motif occurrence in a given sequence). The ZOOPS model takes into consideration the possibility of sequences not containing motifs.

In this case let i be a sequence that does not contain a motif. This extra information is added to our previous model using another parameter λ to denote the prior probability that any position in a sequence is the start of a motif. Next, the probability of the entire sequence to contain a motif is $\lambda = (L - W + 1) * \lambda$

The E-Step

The E-step of the ZOOPS model calculates the expected value of the missing information–the probability that a motif occurrence starts in position j of sequence X_i. The formulas used for the three types of model are given below.

$$Z_{ij}^{t} = \frac{\Pr^{(t)}(X_i | Z_{ij} = 1)\lambda^{(t)}}{\Pr^{(t)}(X_i | Q_i = 0)(1 - \lambda^{(t)}) + \sum_{k=1}^{L-W+1} \Pr^{(t)}(X_i | Z_{ik} = 1)\lambda^{(t)}}$$

where λ^t is the probablity that sequence i has a motif, $Pr^t(X_i|Q_i = 0)$ is the probablity that X_i is generated from a sequence i that does not contain a motif

The M-Step

The M-step of EM in MEME re-estimates the values for λ using the preceding formulas. The math remains the same as for OOPS, we just update the values for λ and γ

$$\lambda^{(t+1)} = \frac{\gamma^{(t+1)}}{(L-W+1)} = \frac{1}{n(L-W+1)} \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j}^{(t)}$$

The model above takes into consideration sequences that do not have any motifs. The challenge is to also take into consideration the situation in which there is more than one motif per sequence. This can be accomplished with the more general model TCM. TCM (two-component mixture model) is based on the assumption that there can be zero, one, or even two motif occurrences per sequence.

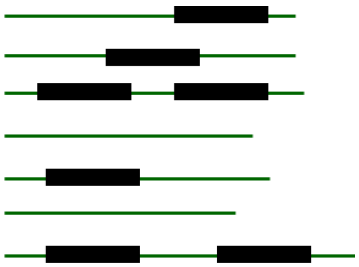


Figure 16.7: Sequences with zero, one or two motifs.

16.10.2 Finding Multiple Motifs

All the above sequence model types model sequences containing a single motif (notice that TCM model can describe sequences with multiple occurences of the same motif). To find multiple, non-overlapping, different motifs in a single dataset, one incorporates information about the motifs already discovered into the current model to avoid rediscovering the same motif. The three sequence model types assume that motif occurrences are equally likely at each position j in sequences x_i. This translates into a uniform prior probability distribution on the missing data variables Z_{ij}. A new prior on each Z_{ij} had to be used during the E-step that takes into account the probability that a new width-W motif occurrence starting at position X_{ij} might overlap occurrences of the motifs previously found. To help compute the new prior on Z_{ij} we introduce variables V_{ij} where $V_{ij} = 1$ if a width-W motif occurrence could start at position j in the sequence X_i without overlapping an occurrence of a motif found on a previous pass. Otherwise $V_{ij} = 0$.

$$V_{\text{ij}} = \begin{cases} 1, & \text{no previous motifs in } [X_{i,j}, ..., X_{i,j+w-1}] \\ 0, & \text{otherwise} \end{cases}$$

16.11 Motif Representation and Information Content

To begin our dicussion, we will talk about uncertainty and probability. Uncertainty is related to our surprise at an event. The event "the sun will rise tomorrow" is not very surprising, so it's uncertainty is quite low. However, the event "the sun will not rise tomorrow" is very surprising and it has a high uncertainty. In general, we can model uncertainty by $-\log p$.

The Shannon entropy is a measure of average uncertainty weighted by the probability of an event occurring. It is a measure of randomness. The logarithm is taken base two. The Shannon entropy is given by the equation:

$$H(X) = -\sum_i p_i \log_2 p_i$$

Entropy is maximum at maximum randomness. For example, if a biased coin is tossed, then the entropy is maximal when the coin is fair, that is the toss is most random.

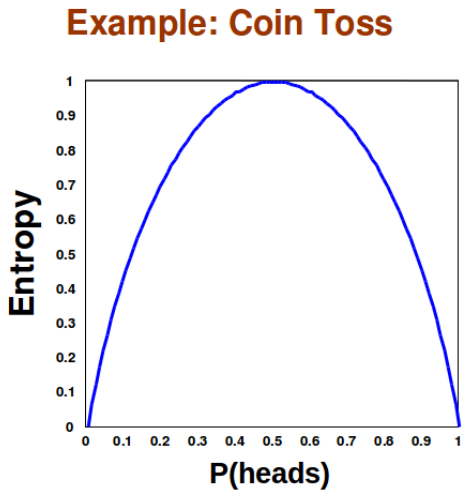


Figure 16.8: Entropy is maximized when the coin is fair, when the toss is most random

Information is a decrease in uncertainty and is measured by the difference in entropy once you receive knowledge about a certain event. We can model a motif by how much uncertainty we lose after applying Gibbs Sampling or EM. In the following figure, the height of each stack represents the number of bits of uncertainty that each position is reduced by. Higher stacks correspond to much certainty about the base at the position of a motif while lower stacks correspond to a higher degree of uncertainty. Specifically, the height of a letter is 2 minus the entropy of that position. The entropy will be high if all bases are weighted equally and low if one base always occurs in one position. Thus, tall stacks correspond to positions where there is a lot of certainty. The height of a letter is proportional to the frequency of the base at that position.

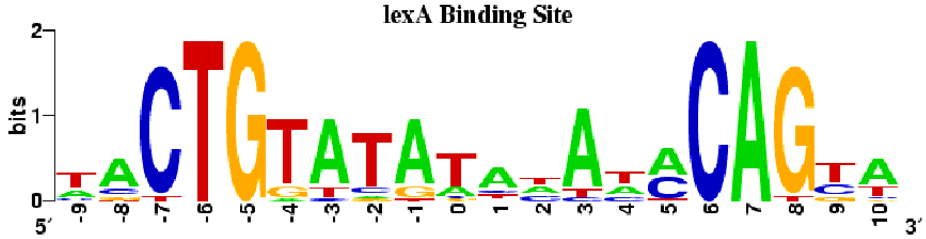


Figure 16.9: The height of each stack represents the number of bits of uncertainty that each position is reduced by.

There is a distance metric on probability distributions known as the Kullback-Leibler distance. This allows us to compare the divergence of the motif distribution to some true distribution. The K-L distance is given by

$D_{KL}(P_{motif}|P_{background}) = \Sigma_{A,T,G,C} P_{motif}(i) \log \frac{P_{motif}(i)}{P_{background}(i)}$

In Plasmodium, there is a lower G-C content. If we assume a G-C content of 20%, then we get the following representation for the above motif. C and G bases are much more unusual, so their prevalence is highly unusual. Note that in this representation, we used the K-L distance, so that it is possible for the stack to be higher than 2.

Figure 16.10: lexA binding site assuming low G-C content and using K-L distance

Bibliography

[1] Timothy L. Bailey. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 28–36. AAAI Press, 1994.

[2] C E Lawrence and A A Reilly. An expectation maximization (em) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins*, 7(1):41–51, 1990.

CHAPTER

SEVENTEEN

REGULATORY GENOMICS

Lecturer: Pouya Kheradpour
Scribe: Maria Frendberg

Figures

17.1 Introduction

Every cell has the same DNA, but they all have different expression patterns due to the temporal and spatial regulation of genes. Regulatory genomics explains these complex gene expression patterns. The regulators we will be discussing are:

- *Transcription Factor (TF)*- Regulates transcription of DNA to mRNA. TFs are proteins which bind to DNA before transcription and either increase or decrease transcription. We can determine the specificity of a TF through experimental methods using protein or antibodies. We can find the genes by their similarity to know TFs.

- *Micro RNA (miRNA)*- Regulates translation of mRNA to Proteins. miRNAs are RNA molecules which bind to mRNA after transcription and can reduce translation. We can determine the specificity of an miRNA through experimental methods, such as cloning, or computational methods, using conservation and structure.

17.1.1 History of the Field

17.1.2 Open Problems

Both TFs and miRNAs are regulators and we can find them through both experimental and computational methods. We will discuss some of these computational methods, specifically the use of evolutionary signatures. These regulators bind to specific patterns, called motifs. We can predict the motifs to which a regulator will bind using both experimental and computational methods. We will be discussing identification of miRNAs through evolutionary and structural signatures and the identification of both TFs and miRNAs through de novo comparative discovery, which theoretically can find all motifs. Given a motif, it is difficult to find the regulator which binds to it.

A target is a place where a factor binds. There are many sequence motifs, however many will not bind; only a subset will be targets. Targets for a specified regulator can be determined using experimental methods. In Lecture 11, methods for finding a motif given a target were discussed. We will also discuss finding targets given a motif.

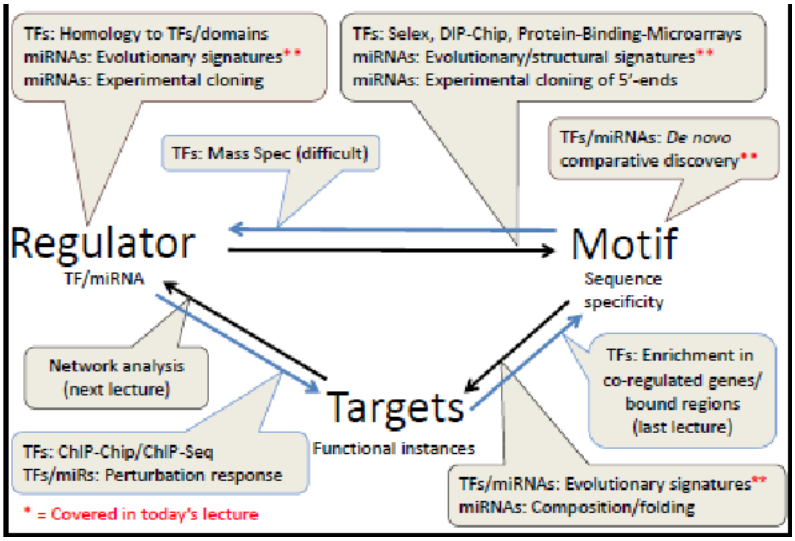


Figure 17.1: Challenges in Regulatory Genomics

17.2 *De Novo* Motif Discovery

17.2.1 TF Motif Discovery

Transcription Factors influence the expression of target genes as either activators or repressors by binding to the DNA near genes. This binding is guided by TF sequence specificity. The closer the DNA is to the base preference, the more likely it is that the factor will bind. These motifs can be found both computationally and experimentally. There are three main approaches for discovering these motifs.

- *Co-Regulation-* In Lecture 11, we discussed a co-regulation type of discovery of motifs by finding sequences which are likely to have the motif bound. We can then use enumerative approaches or

alignment methods to find these motifs in the upstream regions. We can apply similar techniques to experimental data where you know where motif is bound.

- *Factor Centric*- There are also factor centric methods for discovering motifs. These are mostly experimental methods which require a protein or antibody. Examples include SELEX, DIP-Chip, and PBMs. All of these methods are in vitro.

- *Evolutionary*- Instead of focusing on only one factor, evolutionary methods focus on all factors. We can begin by looking at a single factor and determining which properties we can exploit. There are certain sequences which are preferentially conserved (conservation islands). However, these are not always motifs and instead can be due to chance or non-motif conservation. We can then look at many regions, find more conserved motifs, and determine which ones are more conserved overall. By testing conservation in many regions across many genomes, we increase the power. These motifs have certain evolutionary signatures that help us to identify them: motifs are more conserved in intergenic regions than in coding regions, motifs are more likely to be upstream from a gene than downstream. This is a method for taking a known motif and testing if it is conserved.

We now want to find everything that is more conserved than expected. This can be done using a hill climbing approach. We begin by enumerating the motif seeds, which are typically in 3-gap-3 form. Then, each of these seeds is scored and ranked using a conservation ratio corrected for composition and small counts. These seeds are then expanded to fill unspecified bases around the seed using hill climbing. Through these methods, it is possible to arrive at the same, or very similar seeds in different manners. Thus, our final step consists of clustering the seeds using sequence similarity to remove redundancy.

A final method that we can use is recording the frequency with which one sequence is replaced by another in evolution. This produces clusters of k-mers that correspond to a single motif.

17.2.2 Validating Discovered Motifs

There are many ways that we can validate discovered motifs. Firstly, we expect them to match real motifs, which does happen significantly more often than with random motifs. However, this is not a perfect agreement, possibly due to the fact that many known motifs are not conserved and that known motifs are biased and may have missed real motifs. Positional bias. Biased towards TSS,

Motifs also have functional enrichments. If a specific TF is expressed in a tissue, then we expect the upstream region will have that factor's motif. This also reveals modules of cooperating motifs. We also see that most motifs are avoided in ubiquitously expressed genes, so that they are not randomly turned on and off.

17.2.3 Summary

There are disadvantages to all of these approaches. Both TF and region-centric approaches are not comprehensive and are biased. TF centric approaches require a transcription factor or antibody, take lots of time and money, and also have computational challenges. De novo discovery using conservation is unbiased, but it can't match motifs to factors and requires multiple genomes.

17.3 Predicting Regular Targets

17.3.1 Motif Instance Identification

Once potential motifs are discovered, the next step is to discover which motif matches are real. This can be done by both experimental and computational methods.

- *Experimental* - Instances can be identified experimentally using ChIP-Chip and ChIP-Deq methods. Both of these are in vivo methods. This is done by cross linking cells. DNA is first broken into sections. Then the protein and its antibody or tagged protein is added, which binds to various sequences. These bound sequences are now pulled out and cross linking is reversed. This allows us to determine where in the genome the factor was bound. This has a high false positive rate because there are many instances where a factor binds, but is not functional. This is a very popular experimental methods, but it is limited by the availability of antibodies, which are difficult to get for many factors.
- *Computational*- Computation approaches. There are also many computational approaches to identify instances. Single genome approaches use motif clustering. They look for many matches to increase power and are able to find regulatory regions (CRMs). However, they miss instances of motifs that occur alone and require a set of specific factors that act together. Multi-genome approaches, known as phylogentic footprinting, face many challenges. They begin by aligning many sequences, but even in functional motifs, sequences can move, mutate, or be missing. The approach taken by Kheradpour handles this by not requiring perfect conservation (by using a branch length score) and by not requiring an exact alignment (by searching within a window).

Branch Length Scores (BLS) are computed by taking a motif match and searching for it in other species. Then, the smallest subtree containing all species with a motif match is found. The percentage of total tree is the BLS. Calculating the BLS in this way allows for mutations permitted by motif degeneracy, misalignment and movement within a window, and missing motifs in dense species trees.

This BLS is then translated into a confidence score. This enables us to evaluate the likelihood of a given score and to account for differences in motif composition and length. We calculate this confidence score by counting all motif instances and control motifs at each BLS. We then want to see which fraction of the motif instances seem to be real. The confidence score is then signal/(signal+noise). The control motifs used in this calculation are produced by producing 100 shuffles of the original motif, and filtering the results by requiring that they match the genome with +/- 20% of the original motif. These are then sorted based on their similarity to known motifs and clustered. At most one motif is taken from each cluster, in increasing order of similarity, to produce our control motifs.

17.3.2 Validating Targets

Similar to motif discovery, we can validate targets by seeing where they fall in the genome. Confidence selects for TF motif instances in promoters and miRNA motifs in 3' UTRs, which is what we expect. TFs can occur on either strand, whereas miRNA must fall on only one strand. Thus, although there is no preference for TFs, miRNA are found preferentially on the plus strand.

Another method of validating targets is by computing enrichments. This requires having a background and foreground set of regions. These could be a promoter of co-regulated genes vs all genes or regions bound by a factor vs other intergenic regions. Enrichment is computed by taking the fraction of motif

instances inside the foreground vs the fraction of bases in the foreground. Composition and conservation level are corrected for with control motifs. These fractions can be made more conservative using a binomial confidence interval.

Targets can then be validated by comparing to experimental instances found using ChIP-Seq. This shows the conserved CTCF motif instances are highly enriched in ChIP-Seq sites. Increasing confidence also increases enrichment. Using this, many motif instances are verified. ChIP-Seq does not always find functional motifs, so these results can further be verified by comparing to conserved bound regions. This finds that enrichment in intersections is dramatically higher. This shows where factors are binding that have an effect worthwhile conserving in evolution. These two approaches are complementary and are even more effective when used together.

17.4 MicroRNA Genes and Targets

17.4.1 MiRNA Gene Discovery

MiRNAs are post-transcriptional regulators that bind to mRNAs to silence a gene. They are an extremely important regulator in development. These are formed when a miRNA gene is transcribed from the genome. The resulting strand forms a hairpin at some point. This is processed, trimmed and exported to the cytoplasm. Then, another protein trims the hairpin and one half is incorporated into a RISK complex. By doing this, it is able to tell the RISK complex where to bind, which determines which gene is turned off. The second strand is usually discarded. It is a computational problem to determine which strand is which. The computational problem here is how to find the genes which correspond to these miRNAs.

The first problem is finding hairpins. Simply folding the genome produces approximately 760,000 hairpins, but there are only 60 to 200 true miRNAs. Thus we need methods to help improve specificity. Structural features, including folding energy, loops (number, symmetry), hairpin length and symmetry, substructures and pairings, can be considered, however, this only increases specificity by a factor of 40. Thus structure alone cannot predict miRNAs. Evolutionary signatures can also be considered. MiRNA show characteristic conservation properties. Hairpins consist of a loop, two arms and flanking regions. In most RNA, the loop is the most well conserved due to the fact that it is used in binding. In miRNA, however, the arms are more conserved because they determine where the RISK complex will bind. This increases specificity by a factor of 300. Both these structural features and conservation properties can be combined to better predict potential miRNAs.

These features are combined using machine learning, specifically random forests. This produces many weak classifiers (decision trees) on subsets of positives and negatives. Each tree then votes on the final classification of a given miRNA. Using this technique allows us to reach the desired sensitivity (increased by 4,500 fold).

17.4.2 Validating Discovered MiRNAs

Discovered miRNAs can be validated by comparing to known miRNAs. An example given in class shows that 81% of discovered miRNAs were already known to exist, which shows that these methods perform well. The putative miRNAs have yet to be tested, however this can be difficult to do as testing is done by cloning.

Region specificity is another method for validating miRNAs. In the background, hairpins are fairly evenly

distributed between introns, exons, intergenic regions, and repeats and transposons. Increasing confidence in predictions causes almost all miRNAs to fall in introns and intergenic regions, as expected. These predictions also match sequencing reads.

This also produced some genomic properties typical of miRNAs. They have a preference for transcribed strand. This allows them to piggyback in intron of real gene, and thus not require a separate transcription. They also clustering with known and predicted miRNAs. This indicates that they are in the same family and have a common orgin.

17.4.3 MiRNA's 5' End Identification

The first seven bases determine where an miRNA binds, thus it is important to know exactly where clevage occurs. If this clevage point is wrong by even two bases, the miRNA will be predicted to bind to a completely different gene. These clevage points can be discovered computationally by searching for highly conserved 7-mers which could be targets. These 7-mers also correlate to a lack of anti-targets in ubiquitously expressed genes. Using these features, structural features and conservational features, it is possible to take a machine learning approach (SVMs) to predict clevage site. Some miRNAs have no single high scoring position, and these also show imprecise processing in the cell. If the star sequence is highly scored, then it tends to be more expressed in the cell also.

17.4.4 Functional Motifs in Coding Regions

Each motif type has distinct signatures. DNA is strand symmetric, RNA is strand-specific and frame-invariant, and Protein is strand-specific and frame-biased. This frame-invariance can be used as a signature. Each frame can then be evaluated separately. Motifs due to di-codon usage biases are conserved in only one frame offset while motifs due to RNA-level regulation are conserved in all three frame offsets. This allows the ability to distinguish overlapping pressures.

17.5 Current Research Directions

17.6 Further Reading

17.7 Tools and Techniques

17.8 What Have We Learned?

Bibliography

CHAPTER

EIGHTEEN

EPIGENOMICS/CHROMATIN STATES

TODO: missing *@scribe: who was this scribed by?*

Figures

18.1 Introduction

The human body contains approximately 210 different cell types, but each cell type shares the same genomic sequence. In spite of having the same genetic code, cells not only develop into distinct types from this same sequence, but also maintain the same cell type over time and across divisions. This information about the cell type and the state of the cell is called *epigenomic* information. The epigenome ("epi" means above in Greek, so epigenome means above genome) is the set of chemical modifications or marks that influence gene expression and are transferred between generations of cells [6] and between generations of people.

The study of the epigenome is particularly interesting because it is one of the first pieces of evidence that goes against Darwin's description of evolution. The human genome does not change much through the course of one's life and before epigenetics was studied, people thought that the DNA sequence was the sole factor in determining who and what a person is. Epigenetics bring up the interesting possibility that what you do while you are alive – how you live your life – changes your genetic makeup and the genetic makeup of your descendents. Does eating a lot of fatty, greasy foods make your children more susceptible to diabetes? Does habitual smoking change the expression of genes that are important in the respiratory system? These are the kinds of questions that can be posed and answered with epigenetics.

As shown in Figure 18.1, epigenomic information in a cell is encoded in diverse ways. For example, direct methylation of DNA (e.g. at CpG dinucleotides) can alter gene expression. Similarly, positioning of nucleosomes (unit of packing of DNA) determines which parts of DNA are accessible to TFs and other enzymes. Finally, a very powerful way to encoding epigenomic information is through chemical modifications (e.g. methylation, acetylation etc) of histone protein tails.

The 2012 Nobel Prize in Physiology and Medicine was awarded to John Gurdon and Shinya Yamanaka, for work that involved the reprogramming mature cells to become pluripotent. They used different techniques that modified the epigenomic information of the cell and modified the cell into an iPS (induced pluripotent stem cell). This research highlights the importance and applications of epigenetics. These stem cells could be used in the future to grow organs or be injected into people to repair tissue.

In this chapter we will go over histones which carry epigenetic information. We will then look at how we can gather information about histones and identify the location of histones on the genome using CHIP-seq and CHIP-chip. We then see how to analyze this information using the Burrows-Wheeler to allow for efficient search and mapping of our data. From this we then abstract a level and use a hidden Markov model (HMM) to look at chromatin states and group them by function. This will give us an idea of different chromatin states and their function on the human genome.

18.2 Epigenetic Information in Nucleosomes

A key method of encoding epigenetic data is right on the DNA itself. The DNA is annotated with histones that are further chemically modified. You can think of the histones as post-its on the genome. First we will see histones are and how the interact with DNA. Then we will look at how they can signal things and affect gene expression.

In order to fit two meters of DNA into a 5-20 μm diameter cell nucleus and arrange the DNA for easy access to transcriptional machinery, DNA is packaged into chromatin. Nucleosomes form the unit of this

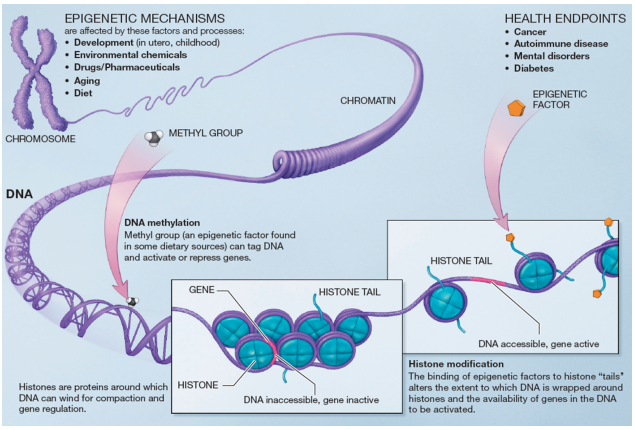


Figure 18.1: Types of epigenetic modifications

packaging. A nucleosome is composed of DNA approximately 150-200 bp long wrapped around a histone protein octamer consisting of two copies each of histone proteins H2A, H2B, H3, and H4 (and occasionally a linker histone H1 or H5). While the structure and importance of higher-level packaging of nucleosomes is less known, the lower-level arrangement and modification of nucleosomes is very important to transcriptional regulation and the development of different cell types. Histone proteins H3 and H4 are the most highly conserved proteins in the eukaryotic domain of life. If DNA contains the blueprints of life, nucleosomes contain the blueprints of life with multiple cell types.

Nucleosomes encode epigenetic information in two ways. First, their positions on the DNA are determine which parts of DNA are accessible. Nucleosomes are often bound to the promoters of inactive genes. To initiate transcription of a gene, transcription factors (TFs) and the General Factors have to bind to its promoter. Therefore, when a gene becomes active, the nucleosomes located on its promoter are often pushed aside or removed. The promoter will remain exposed until further modifications are made. Hence, nucleosome positioning on the DNA is stable, yet mutable. This property of stability and mutability is a prerequisite for any form of epigenetic information because cells need to maintain the identity of a particular cell type, yet still be able to change their epigenetic state to respond to environmental circumstances

Second, nucleosomes contain tails of amino acid residues protruding from the ends of their histones. These tails can undergo post-translational modification such as methylation, acetylation and phosphorylation which affect gene expression by recruiting initiation factors. There are over 100 distinct histone modifications that have been found experimentally. This has lead to the "histone code hypothesis" that combinations of chromatin modifications encode biological function. These modifications are so common that a shorthand has been developed to identify them. This shorthand consists of the name of the histone protein, the amino acid residue on its tail that has been modified and the type of modification made to this residue. To illustrate, the fourth residue from the N-terminus of histone H3, lysine, is often methylated at the promoters of active genes. This modification is described as H3K4me3 (if methylated thrice). The first part of the shorthand corresponds to the histone protein, in this case H3; K4 corresponds to the 4th residue from the end, in this case a lysine, and me3 corresponds to the actual modification, the addition of 3 methyl groups in this case.

18.3 Technologies for measurement of epigenetic signals

Given the importance of epigenomic information in biology, great efforts have been made to study these signals on DNA. One common method for epigenomic mark measurement is chromatin immunoprecipitation (ChIP). The procedures of ChIP are described as follows and depicted in Figure 18.2:

1. Cells are exposed to a cross-linking agent such as formaldehyde, which causes covalent bonds to form between DNA and its bound proteins (e.g. histones with specific modifications);

2. Genomic DNA is isolated from the cell nucleus;

3. Isolated DNA is sheared by sonication or enzymes;

4. Antibodies against a specific epigenetic mark are then added to pull out its associated DNA. These antibodies are generated by exposing proteins of interest to mammals (e.g. goats or rats). The resulting immune response will cause the production of specific antibodies.

5. The cross-linking between the protein and DNA is reversed and the DNA fragments specific to the epigenetic marks are purified.

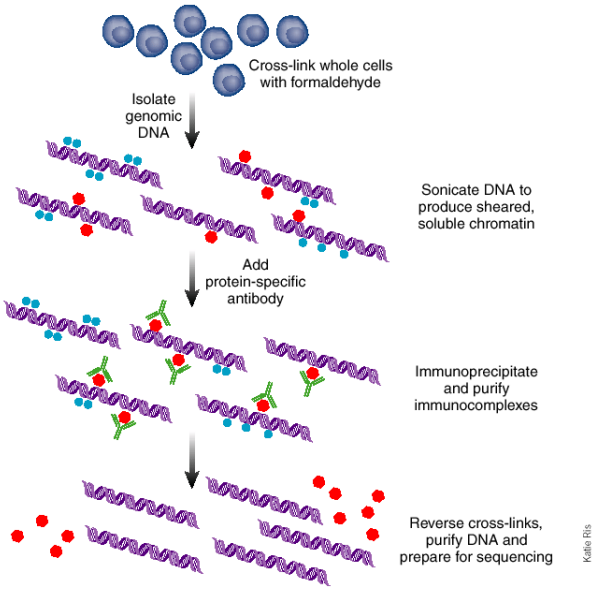


Figure 18.2: Chromatin immunoprecipitation [5]

So after this process we have different short sequences of DNA that correspond to spots where histones were bound to the DNA. To identify the location of these DNA fragments on the genome, one can hybridize them to known DNA segments on an array or gene chip and visualize them with fluorescent marks (called ChIP-chip). Alternatively, one can do massive parallel next-generation sequencing of these fragments (called ChIP-seq). Each sequence tag is 30 base pairs long. These tags are mapped to unique positions in the reference genome of 3 billion bases. The number of reads depending on sequencing depth, but typically there are on the order of 10 million mapped reads for each ChIP-seq experiment. ChIP-seq is preferred over ChIP-chip nowadays because it has wider dynamic range of detection and can avoid problems such as cross-hybridization in ChIP-chip. Given this data from ChIP-chip or CHIP-seq, analysis of epigenetic data consists of various steps. First, the reads from ChIP must be mapped to the DNA (called read mapping). Next, we must determine which readings correspond to presence of a chromatin mark (called peak calling). After these preprocessing steps, we can build different supervised and unsupervised models to study chromatin states and their relation to biological function. We look at each of these steps in turn.

18.4 Read Mapping

The problem of read mapping seeks to assign a given read to the best matching location in the reference genome. Given the vast amount of reads and the size of human genome, one common requirement of all read mapping algorithms is the algorithmic time efficiency. We are going to have a huge number of reads and we want to place each of them on the genome which is billions of base pairs long, we need a very fast algorithm to allow us to do this in a reasonable time.

Based on previous lectures, we can image different ways to perform mapping of reads - sequence alignment (O(mn)), hash-based approaches such as MAQ, linear time string matching (O(m+n)) and suffix trees and arrays (O(m)). However, a problem with all these techniques is that they have a large memory requirement (e.g. O(mn)). Instead, state-of-the-art techniques based on the Burrows-Wheeler transformation [1] runs in O(m) time and require O(n) space.

The Burrows-Wheeler transform came from the need to compress information. It would take a long string and rearrange it in a way that there were often repeating letters. This string could be compressed because instead of writing 100 A's the computer could just indicate that there were now 100 A's in a row. The Burrows-Wheeler transform also has some other special properties that we will be exploiting to search in sublinear time.

The Burrows-Wheeler transform creates a unique transformed string that is shorter than the original string. It also can be reversed easily to generate your original string, so no information is lost. The transformed string is in sorted order which allows for easy searching. The details of Burrows-Wheeler transformation are described as follows and illustrated in Figure 18.3:

Input	All Rotations	Sorted List of Rotations	Output Last Column
^BANANA@	^BANANA@ @^BANANA A@^BANAN NA@^BANA ANA@^BAN NANA@^BA ANANA@^B BANANA@^	ANANA@^**B** ANA@^BA**N** A@^BANA**N** BANANA@^ NANA@^B**A** NA@^BAN**A** ^BANANA@ @^BANAN**A**	BNN^AA@A

Figure 18.3: The Burrows-Wheeler forward transformation

Add 1	Sort 1	Add 2	Sort 2	Add 3	Sort 3	Add 4	Sort 4	Add 5	Sort 5	Add 6	Sort 6	Add 7	Sort 7	Add 8	Sort 8
B	A	BA	AN	BAN	ANA	BANA	ANAN	BANAN	ANANA	BANANA	ANANA@	BANANA@	ANANA@^	BANANA@^	ANANA@^B
N	A	NA	AN	NAN	ANA	NANA	ANA@	NANA@	ANA@^	NANA@^	ANA@^B	NANA@^B	ANA@^BA	NANA@^BA	ANA@^BAN
N	A	NA	A@	NA@	A@^	NA@^	A@^B	NA@^B	A@^BA	NA@^BA	A@^BAN	NA@^BAN	A@^BANA	NA@^BANA	A@^BANAN
^	B	^B	BA	^BA	BAN	^BAN	BANA	^BANA	BANAN	^BANAN	BANANA	^BANANA	BANANA@	^BANANA@	BANANA@^
A	N	AN	NA	ANA	NAN	ANAN	NANA	ANANA	NANA@	ANANA@	NANA@^	ANANA@^	NANA@^B	ANANA@^B	NANA@^BA
A	N	AN	NA	ANA	NA@	ANA@	NA@^	ANA@^	NA@^B	ANA@^B	NA@^BA	ANA@^BA	NA@^BAN	ANA@^BAN	NA@^BANA
@	^	@^	^B	@^B	^BA	@^BA	^BAN	@^BAN	^BANA	@^BANA	^BANAN	@^BANAN	^BANANA	@^BANANA	^BANANA@
A	@	A@	@^	A@^	@^B	A@^B	@^BA	A@^BA	@^BAN	A@^BAN	@^BANA	A@^BANA	@^BANAN	A@^BANAN	@^BANANA

Figure 18.4: The Burrows-Wheeler reverse transformation

First, producing a transform from an original string consists of the following steps:

1. For a given reference genome, add a special character at the beginning and end of the string (e.g. BANANA becomes `^BANANA@`). Then generate all the rotations of this string, e.g., `NANA@^BA`.
2. Sort the rotations lexicographically, i.e. in alphabetical order, with special characters sorted first.
3. The last column of the sorted list of rotations contains the transformed string

Once a Burrows-Wheeler transform has been computed, we can reverse the transform to compute the original string via the procedure in (Figure 18.4). Briefly, the reverse transformation works as follows: given the transformed string, sort the string characters in alphabetical order; this gives the first column in the transform. Combine these two columns to get pairs of characters. Sort the pairs and repeat.

From the Burrows-Wheeler transform we observe that all occurrences of the same suffix are effectively next to each other rather than scattered throughout the genome. Moreover, the i^{th} occurrence of a character in the first column corresponds to the i^{th} occurrence in the last column. Searching for substrings using the transform is also easy. Suppose we are looking for the substring ANA in the given string. Then the problem of search is reduced to searching for a prefix ANA among all possible sorted suffixes (generated by rotations). In the case of read mapping, the genome is transformed using the Burrows-Wheeler transform and stored in O(n) space. We can then efficiently search for substrings (reads) using the strategy described above.

18.5 Peak Calling

After reads are aligned, signal tracks as shown in Figure 18.5 can be computed. This data can be ordered into a long histogram spanning the length of the genome and indicating the number of reads (or degree of fluorescence in the case of ChIP-chip) found at each position in the genome. More reads (or fluorescence) means the epigenomic marker of interest is most often present at this particular location. For histone modifications, peak calling methods are based on univariate Hidden Markov Model, or scan statistics (count the reads within bins of certain size and apply statistical analysis). Problems tend to arise with broader domains due to the ambiguity of calling them one large peak or multiple smaller peaks. Instead, a simple but effective strategy consists of *Data binarization*. During data binarization, the only call to be made is whether a chromatin marker is observed in an interval or not. Typically, one can set a threshold on the amount of signal that must be present and then any interval with more than that quantity of signal gets a value of 1 and gets 0 otherwise. Data binarization makes the data easy to interpret, less prone to overfitting and amenable to simpler models.

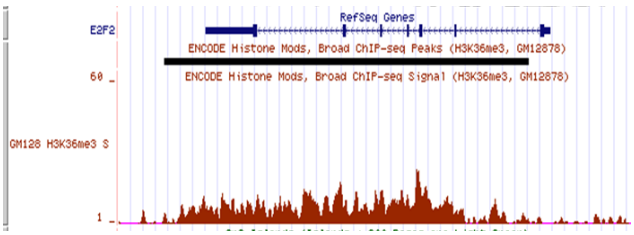


Figure 18.5: Sample signal tracks

We now move on to techniques for interpreting chromatin marks. There are many ways to analyze epigenomic marks such as aggregating chromatin signals (such as H3K4me3) on known feature types (e.g. promoters of genes with high or low expression levels), performing supervised or unsupervised machine

learning methods to derive epigenomic features that are predictive of different types of genomics elements such as promoters, enhancers or large intergenic non-coding RNAs. In particular, in this lecture, we examine in detail the analysis of chromatin marks as done in [3].

18.6 Annotating the Genome Using Chromatin Signatures

The histone code hypothesis suggests that chromatin-DNA interactions are guided by **combinatorial histone modifications**. These combinatorial modifications, when taken together, can in part determine how a region of DNA is interpreted by the cell (i.e. as a transcription factor binding domain, a splice site, an enhancer region, an actively expressed gene, a repressed gene, or a non functional region). We are interested in interpreting this "code" (i.e. determining from histone marks at a region whether the region is a transcription start site, enhancer, promoter, etc.). With an understanding of the combinatorial histone marks, we can annotate the genome into functional regions and predict novel enhancers, promoters, genes, etc. The challenge is that there are dozens of marks and they exhibit complex combinatorial effects.

In this section, we explore a technique for interpreting the "code" and its application to a specific dataset [3], which measured 41 chromatin marks across the human genome.

18.6.1 Data

Data for this analysis consisted of 41 chromatin marks including acetylations, methylations, H2AZ, CTCF and PolII in CD4 T cells. First, the genome was divided into 200 bp non-overlapping bins in which the binary absence or presence of each of the 41 chromatin marks was determined. This data was processed using **data binarization** as described in the previous section. Specifically, let C_{ij} be the number of reads detected by ChIP-seq for mark i, mapping to the 200bp bin j. Let λ_i be the average number of reads mapping to a bin for mark i. The mark i is determined to be present in bin j if $P(X > C_{ij}) < 10^{-4}$ **TODO: expand** *@scribe: How is the threshold chosen?* where X is a Poisson random variable with mean λ_i and absent otherwise. In order words, the read enrichment for a specific bin has to be significantly greater than a random process of putting reads into bins. An example for chromatin states around the CAPZA2 gene on chromosome 7 is shown in Figure 18.6. So in this way, for each mark i, we can label each bin j with a 1 if the mark is present and a 0 if it isn't. Looking at the data as a whole, we can think of it as large binary matrix, where each row corresponds to a mark and each column corresponds to a bin (which is simply a 200bp region of the genome).

Additional data used for analysis included gene ontology data, SNP data, expression data, and others.

18.6.2 HMMs for Chromatin State Annotation

Our goal is to identify biologically meaningful and spatially coherent combinations of chromatin marks. Remember that we broke the genome up into 200bp blocks, so by spatially coherent we mean that if we have a genomic element that is longer than 200bps, we expect the combination of chromatin marks to be consistent on each 200bp bin in the region. We'll call these biologically meaningful and spatially coherent combinations of chromatin marks **chromatin states**. In previous lectures, we've seen HMMs applied to genome annotation for genes and CpG islands. We would like to apply the same ideas to this situation, but in this case, we don't know the hidden states a priori (e.g. CpG island region or not), we'd like to learn them *de novo*. This model can capture both the functional ordering of different states (e.g from promoter

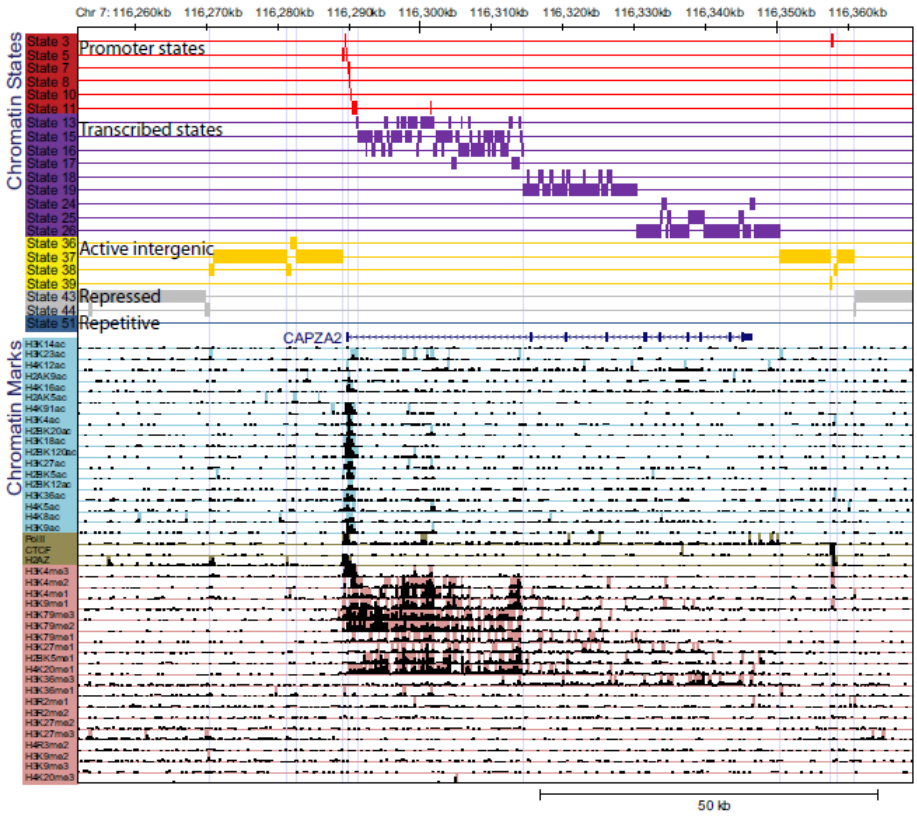


Figure 18.6: Example of the data and the annotation from the HMM model. The bottom section shows the raw number of reads mapped to the genome. The top section shows the annotation from the HMM model.

to transcribed regions) and the spreading of certain chromatin domains across the genomes. To summarize, we want to learn an HMM where the hidden states of the HMM are chromatin states.

As we learned previously, even if we don't know the emission probabilities and transition probabilities of an HMM, we can use the Baum-Welch training algorithm to learn the maximum likelihood values for those parameters. In our case, we have an added difficulty, we don't even know how many chromatin states exist! In the following subsections, we'll expand on how the data is modeled and how we can choose the number of states for the HMM.

Emission of a Vector

In HMMs from previous lectures, each state emitted either a single nucleotide or a single string of nucleotides at a time. In the HMM for this problem, each state emits a combination of epigenetic marks. Each combination can be represented as an n-dimensional vector where n is the number of chromatin marks being analyzed ($n = 41$ for our data). For example, assuming you have four possible epigenetic modifications: H3K4me3, H2BK5ac, Methyl-C, and Methyl-A, a sequence containing H3K4me3 and Methyl-C could be presented as the vector (1, 0, 1, 0). One could imagine many different probability distributions on binary n-vectors and for simplicity, we assume that the marks are independent and modeled as Bernoulli random variables. So we are assuming the marks are independent given the hidden state of the HMM (note that this is not the same as assuming the marks are independent).

If there are n input marks, each state k has a vector $(p_{k1}, .., p_{kn})$ of probabilities of observing marks 1 to n. Since the probability is modeled as a set of independent Bernoulli random variables, the probability of observing a set of marks given that we are in the hidden state k equals the product of the probabilities of observing individual marks. For example if $n = 4$, the observed marks at bin j were $(1, 0, 1, 0)$ and we were in state k, then the likelihood of that data is $p_{k1}(1 - p_{k2})p_{k3}(1 - p_{k4})$.

The learned emission probabilities for the data are shown in Figure 18.7.

Figure 18.7: Emission probabilities for the final model with 51 states. The cell corresponding to mark i and state k represents the probability that mark i is observed in state k.

Transition Probabilities

Recall that the transition probabilities represent the frequency of transitioning from one hidden state to another hidden state. In this case, our hidden states are chromatin states. The transition matrix for our data is shown in Figure 18.8. As seen from the figure, the matrix is sparse, indicating that only a few of the possible transitions actually occur. The transition matrix reveals the spatial relationships between neighboring states. Blocks of states in the matrix reveal sub-groups of states and from these higher level blocks, we can see transitions between these meta-states.

18.6.3 Choosing the Number of states to model

As with most machine learning algorithms, increasing the complexity of the model (e.g. the number of hidden states) will allow it to better fit training data. However, the training data is only a limited sample of

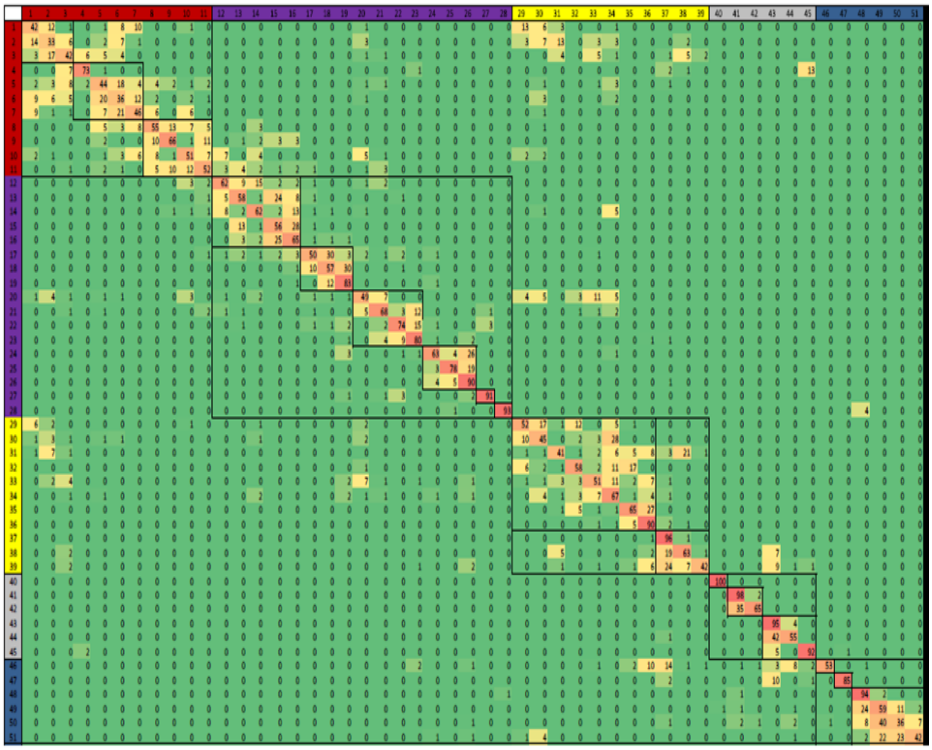


Figure 18.8: Transition probabilities for the final model with 51 states. The transition probability increases from green to red. Spatial relationships between neighboring chromatin states and distinct sub-groups of states are revealed by clustering the transition matrix. Notably, the matrix is sparse, so indicating that most are not possible.

the true population. As we add more complexity, at some point we are fitting patterns in the training data that only exist due to limited sampling, so that the model will not generalize to the true population. This is called **over-fitting** training data; we should stop adding complexity to the model before it fits the noise in the training data.

Bayesian Information Criterion (BIC) is a common technique for optimizing the complexity of a model that balances increased fit to the data with complexity of the model. Using BIC, we can visualize the increasing power of the HMM as a function of the number of states. Generally, one will choose a value for k (the number of states) such that the addition of more states has relatively little benefit in terms of predictive power gain. However, there is a tradeoff between model complexity and model interpretability that BIC cannot help with. The optimal model according to BIC is likely to have more states than an ideal model because we are willing to trade some predictive power for a model with fewer states that can be interpreted biologically. The human genome is so big and the chromatin marks so complex that statistically significant differences are easy to find, yet many of these differences are not biologically significant.

To solve this problem, we start with a model with more hidden states than we believe are necessary and prune hidden states as long as all states of interest in the larger model are adequately captured. The Baum-Welch algorithm (and EM in general) is sensitive to the initial conditions, so we try several random initializations in our learning. For each number of hidden states from 2 - 80, we generate three random initializations of the parameters and train the model using Baum-Welch. The best model according to BIC had 79 states and states were then iteratively removed from this set of 79 states.

As we mentioned earlier, Baum-Welch is sensitive to the initial parameters, so when we pruned states,

we used a nested initialization rather than a randomized initialized for the pruned model. Specifically, states were greedily removed from the BIC-optimal 79 state model. The state to be removed was the state that such that all states from the 237 randomly initialized models were well captured. When removing a state, the emission probabilities would be removed and any state transitioning to the removed state would have that transition probability uniformly redistributed to the remaining states. This was used as the initialization to the Baum-Welch training. The number of states for a model to analyze can then be selected by choosing the model trained from such nested initialization with the smallest number of states that sufficiently captures all states offering distinct biological interpretations. The resulting final model had 51 states.

We can also check model fit by looking at how the data violates model assumptions. Given the hidden state, the HMM assumes that each mark is independent. We can test how well the data conforms to this assumption by plotting the dependence between marks. This can reveal states that fit well and those that do not. In particular, repetitive states reveal a case where the model does not fit well. As we add more states, the model is better able to fit the data and hence fit the dependencies. By monitoring the fit on individual states that we are interested in, we can control the complexity of the model.

18.6.4 Results

This multivariate HMM model resulted in a set of 51 biologically relevant chromatin states. However, there were no one-to-one relationship between each state and known classes of genomic elements (e.g. introns, exons, promoters, enhancers, etc) Instead, multiple chromatin states were often associated with one genomic element. Each chromatin state encoded specific biological relevant information about its associated genomic element. For instance, three different chromatin states were associated with transcription start site (TSS), but one was associated with TSS of highly expressed genes, while the other two were associated with TSS of medium and lowly expressed genes respectively. Such use of epigenetic markers greatly improved genome annotation, particularly when combined with evolutionary signals discussed in previous lectures. The 51 chromatin states can be divided in five large groups. The properties of these groups are described as follows and further illustrated in 18.9:

1. **Promoter-Associated States (1-11):**

   These chromatin states all had high enrichment for promoter regions. 40-89% of each state was within 2 kb of a RefSeq TSS. compared to 2.7% genome-wide. These states all had a high frequency of H3K4me3, significant enrichments for DNase I hypersensitive sites, CpG islands, evolutionarily conserved motifs and bound transcription factors. However, these states differed in the levels of associated marks such as H3K79me2/3, H4K20me1, acetylations etc. These states also differed in their functional enrichment based on Gene Ontology (GO). For instance, genes associated with T cell activation were enriched in state 8 while genes associated with embryonic development were enriched in state 4. Additionally, among these promoter states there were distinct positional enrichments. States 1-3 peaked both upstream and downstream of TSS; states 4-7 were concentrated right over TSS whereas states 8-11 peaked between 400 bp and 1200 bp downstream of TSS. This suggests that chromatin marks can recruit initiation factors and that the act of transcript can reinforce these marks. The distinct functional enrichment also suggests that the marks encode a history of activation.

2. **Transcription-Associated States (12-28):**

   This was the second largest group of chromatin states and included 17 transcription-associated states. There are 70-95% contained in annotated transcribed regions compared to 36% for rest of genome. These states were not predominantly associated with a single mark but rather they were defined by a combination of seven marks - H3K79me3, H3K79me2, H3K79me1, H3K27me1, H2BK5me1, H4K20me1 and H3K36me3. These states have subgroups associated with 5'-proximal or 5'-distal locations. Some of these states were associated with spliced exons, transcription start sites or end sites. Of interest,

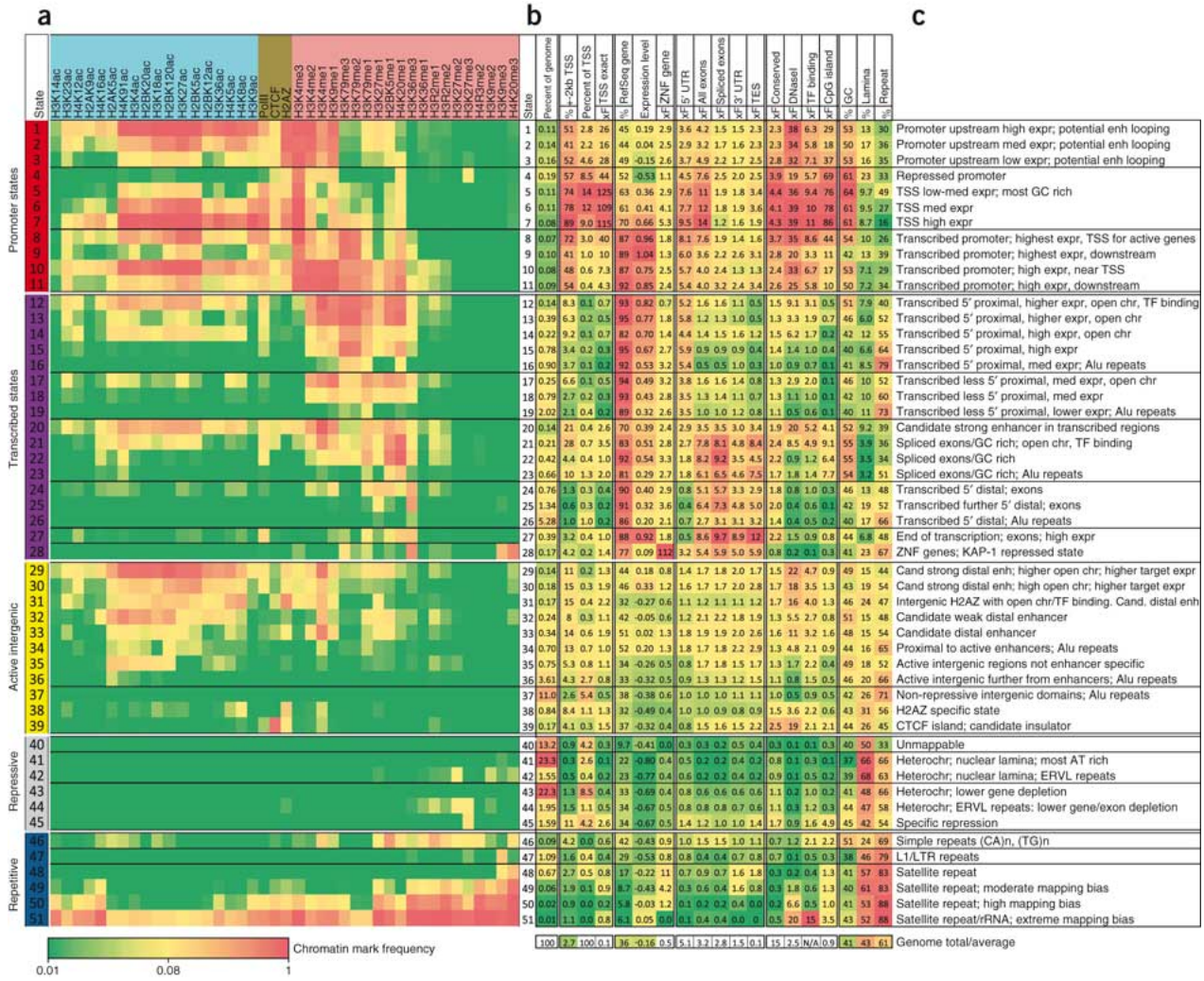


Figure 18.9: Chromatin state definition and functional interpretation. [3] a. Chromatin mark combinations associated with each state. Each row shows the specific combination of marks associated with each chromatin state and the frequencies between 0 and 1 with which they occur in color scale. These correspond to the emission probability parameters of the HMM learned across the genome during model training. b. Genomic and functional enrichments of chromatin states, including fold enrichment in different part of the genome (e.g. transcribed regions, TSS, RefSeq 5 end or 3end of the gene etc), in addition to fold enrichment for evolutionarily conserved elements, DNaseI hypersensitive sites, CpG islands, etc. All enrichments are based on the posterior probability assignments. c. Brief description of biological state function and interpretation (chr, chromatin; enh, enhancer).

state 28, which was characterized by high frequency for H3K9me3, H4K20me3, and H3K36me3, showed a high enrichment in zinc-finger genes. This specific combination of marks was previously reported as marking regions of KAP1 binding, a zinc-finger specific co-repressor.

3. **Active Intergenic States (29-39):**

These states were associated with several classes of candidate enhancer regions and insulator regions and were associated with higher frequencies for H3K4me1, H2AZ, several acetylation marks but lower frequencies of methylation marks. Moreover, the chromatin marks could be used to distinguish active from less active enhancers. These regions were usually away from promoters and were outside of transcribed genes. Interestingly, several active intergenic states showed a significant enrichment for disease

SNPs, or single nucleotide polymorphism in genome-wide association study (GWAS). For instance, a SNP (rs12619285) associated with plasma eosinophil count levels in inflammatory diseases was found to be located in the chromatin state 33, which was enriched for GWAS hits. In contrast, the surrounding region of this SNP was assigned to other chromatin states with no significant GWAS association. This can shed light on the possible functional significance of disease SNPs based on its distinct chromatin states.

4. **Large-Scale Repressed States (40-45):**

   These states marked large-scale repressed and heterochromatic regions, representing 64% of the genome. H3K27me3 and H3K9me3 were two most frequently detected marks in this group.

5. **Repetitive States (46-51):**

   These states showed strong and distinct enrichments for specific repetitive elements. For instance, state 46 had a strong sequence signature of low-complexity repeats such as (CA)n, (TG)n, and (CATG)n. States 48-51 showed seemingly high frequencies for many modification but also enrichment in reads from non-specific antibody control. The model was thus able to also capture artifacts resulting from lack of coverage for additional copies of repeat elements.

Since many of the chromatin states were described by multiple marks, the contribution of each mark to a state was quantified. Varying subsets of chromatin marks were tested to evaluate their potential for distinguishing between chromatin states. In general, increasing subsets of marks were found to converge to an accurate chromatin state when marks were chosen greedily.

The predictive power of chromatin states for discovery of functional elements consistently outperformed predictions based on individual marks. Such unsupervised model using epigenomic mark combination and spatial genomic information performed as well as many supervised models in genome annotation. It was shown that this HMM model based on chromatin states was able to reveal previously unannotated promoters and transcribed regions that were supported by independent experimental evidence. When chromatin marks were analyzed across the whole genome, some of the properties observed were satellite enriched states (47-51) enriched in centromere, the zinc-finger enriched state (state 28) enriched on chromosome 19 etc. Thus, such genome-wide annotation based on chromatin states can help better interpret biological data and potentially discover new classes of functional elements in the genome.

18.6.5 Multiple Cell Types

All of the above work was done in a single cell type (CD4+ T cells). Since epigenomic markers vary over time, across cell types, and environmental circumstances, it is important to consider the dynamics of the chromatin states across different cell types and experimental conditions. The ENCODE project [2] in the Brad Bernstein Chromatin Group has measured 9 different chromatin marks in nine human cell lines. In this case, we want to a learn a single set of chromatin marks for all of the data. There are two approaches to this problem: concatenation and stacking. For concatenation, we could combine all of the 9 cell lines as if they were a single cell line. By concatenating the different cell lines, we ensure that a common set of state definitions are learned. We can do this here because the profiled marks were the same in each experiment. However, if we profiled different marks for different cell lines, we need to use another approach. Alternatively, we can align the 9 cell lines and treat all of the marks as a super-vector. This allows us to learn cell line specific activity states, for example there might be a state for ES-specific enhancers (in that state there would be enhancer marks in ES, but no marks in other cell types). Unfortunately, this greatly increases the dimension of the vectors emitted by the HMM, which translates to an increase in the model complexity needed to adequately fit the data.

Suppose we had multiple cell types where we profiled different marks and we wanted to concatenate them. One approach is to learn independent models and then combine them. We could find corresponding states by matching emission vectors that are similar or by matching states that appear at the same places in the genome. A second approach is to treat the missing marks as missing data. The EM framework allows for unspecified data points, so as long as pairwise relationships are observed between marks in some cell type, we can use EM. Lastly, we can predict the missing chromatin marks based on the observed marks using maximum-likelihood as in the Viterbi algorithm. This is a less powerful approach if the ultimate goal is chromatin state learning because we are only looking at the most likely state instead of averaging over all possibilities as in the second approach.

In the case with 9 marks in 9 hyman cell lines, the cell lines were concatenated and a model with 15 states was learned [4]. Each cell type was analyzed for class enrichment. It was shown that some chromatin states, such as those encoding active promoters were highly stable across all cell types. Other states, such as those encoding strong enhancers, were highly enriched in a cell-type specific manner, suggesting their roles in tissue specific gene expression. Finally, it was shown that there was significant correlation between the epigenetic marks on enhancers and the epigenetic marks on the genes they regulate, even though these can be thousands of base pairs away. Such chromatin state model has proven useful in matching enhancers to their respective genes, a problem that has been largely unsolved in modern biology. Thus, chromatin states provide a means to study the dynamic nature of chromatin across many cell types. In particular, we can see the activity of a particular region of the genome based on the chromatin annotation. It also allows us to summarize important information contained in 2.4 billion reads in just 15 chromatin states.

18.7 Current Research Directions

Several large-scale data production efforts such as ENCODE, modENCODE and Epigenome Radmap projects are currently in progress and therefore there are several opportunities to computationally analyze this new data.

Epigenomic data is also being used to study how behavior can alter your genome. There are studies being done that look at diet and exercise and their effects on disease susceptibility.

18.8 Further Reading

There are several interesting papers that are looking at chromatin states and epigenetics in general. I have listed several urls below to begin your exploration:

1. http://www.nature.com/nmeth/journal/v8/n9/full/nmeth.1673.html
2. http://www.nature.com/nature/journal/v473/n7345/full/nature09906.html
3. http://www.nature.com/nbt/journal/v28/n8/abs/nbt.1662.html
4. http://www.nytimes.com/2012/09/09/opinion/sunday/why-fathers-really-matter.html?_r=1

18.9 Tools and Techniques

ChromHMM is the HMM described in the text. It is available free for download with instructions and examples at: http://compbio.mit.edu/ChromHMM/.

Segway is another method for analyzing multiple tracks of functional genomics data. It uses a dynamic Bayesian network (HMMs are a particular type of dynamic Bayesian network) which enables it to analyze the entire genome at 1-bp resolution. The downside is that it is much slower than ChromHMM. It is available free for download here: http://noble.gs.washington.edu/proj/segway/.

18.10 What Have We Learned?

In this lecture, we learnt how chromatin marks can be used to infer biologically relevant states. The analysis in [3] presents a sophisticated method to apply previously learnt techniques such as HMMs to a complex problem. The lecture also introduced the powerful Burrows-Wheeler transform that has enabled efficient read mapping.

Bibliography

[1] Langmead B, C Trapnelli, M Pop, and S Salzberg. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biology*, 10(3), 2009.

[2] Encyclopedia of dna elements.

[3] J. Ernst and M. Kellis. Discovery and characterization of chromatin states for systematic annotation of the human genome. *Nature Biotechnology*, 28:817–825, 2010.

[4] J. Ernst, P. Kheradpour, T.S. Mikkelsen, N. Shoresh, L.D. Ward, C.B. Epstein, X. Zhang, L. Wang, R. Issner, M. Coyne, et al. Mapping and analysis of chromatin state dynamics in nine human cell types. *Nature*, 473(7345):43–49, 2011.

[5] Elaine R Mardis. Chip-seq: welcome to the new frontier. *Nat Meth*, 4(8):614–614, 2007.

[6] Ncbi nih epigenomics.

CHAPTER

NINETEEN

REGULATORY NETWORKS: INFERENCE, ANALYSIS, APPLICATION

Guest Lecture by
Sushmita Roy (2010) / Soheil Feizi (2012)
Scribed by Ben Holmes (2010) / Hamza Fawzi and Sara Brockmueller (2012)

Figures

19.1 Introduction

Living systems are composed of multiple layers that encode information about the system. The primary layers are:

1. Epigenome: Defined by chromatin configuration. The structure of chromatin is based on the way that histones organize DNA. DNA is divided into nucleosome and nucleosome-free regions, forming its final shape and influencing gene expression. [1]

[1] More in the epigenetics lecture.

2. Genome: Includes coding and non-coding DNA. Genes defined by coding DNA are used to build RNA, and Cis-regulatory elements regulate the expression of these genes.
3. Transcriptome RNAs (ex. mRNA, miRNA, ncRNA, piRNA) are transcribed from DNA. They have regulatory functions and manufacture proteins.
4. Proteome Composed of proteins. This includes transcription factors, signaling proteins, and metabolic enzymes.

Interactions between these components are all different, but understanding them can put particular parts of the system into the context of the whole. To discover relationships and interactions within and between layers, we can use networks.

19.1.1 Introducing Biological Networks

Biological networks are composed as follows:

Regulatory Net – set of regulatory interactions in an organism.

- Nodes are regulators (ex. transcription factors) and associated targets.
- Edges correspond to regulatory interaction, directed from the regulatory factor to its target. They are signed according to the positive or negative effect and weighted according to the strength of the reaction.

Metabolic Net – connects metabolic processes. There is some flexibility in the representation, but an example is a graph displaying shared metabolic products between enzymes.

- Nodes are enzymes.
- Edges correspond to regulatory reactions, and are weighted according to the strength of the reaction.

Signaling Net – represents paths of biological signals.

- Nodes are proteins called signaling receptors.
- Edges are transmitted and received biological signals, directed from transmitter to receiver.

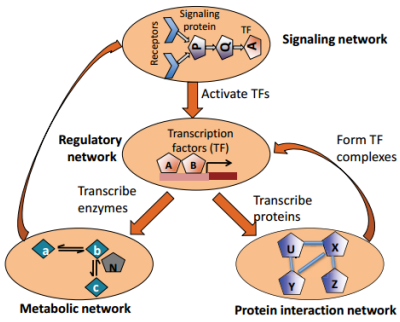
Protein Net – displays physical interactions between proteins.

- Nodes are individual proteins.
- Edges are physical interactions between proteins.

Co-Expression Net – describes co-expression functions between genes. Quite general; represents functional rather than physical interaction networks, unlike the other types of nets. Powerful tool in computational analysis of biological data.

- Nodes are individual genes.
- Edges are co-expression relationships.

Today, we will focus exclusively on regulatory networks. Regulatory networks control context-specific gene expression, and thus have a great deal of control over development. They are worth studying because they are prone to malfunction and causing disease.



(a) Interactions between biological networks.

19.1.2 Interactions Between Biological Networks

Individual biological networks (that is, layers) can themselves be considered nodes in a larger network representing the entire biological system. We can, for example, have a signaling network sensing the environment governing the expression of transcription factors. In this example, the network would display that TFs govern the expression of proteins, proteins can play roles as enzymes in metabolic pathways, and so on.

The general paths of information exchange between these networks are shown in figure 19.2.

19.1.3 Studying Regulatory Networks

In general, networks are used to represent dependencies among variables. Structural dependencies can be represented by the presence of an edge between nodes - as such, unconnected nodes are conditionally independent. Probabilistically, edges can be assigned a "weight" that represents the strength or the likelihood of the interaction. Networks can also be viewed as matrices, allowing mathematical operations. These frameworks provides an effective way to represent and study biological systems.

These networks are particularly interesting to study because malfunctions can have a large effect. Many diseases are caused by rewirings of regulatory networks. They control context specific expression in development. Because of this, they can be used in systems biology to predict development, cell state, system state, and more. In addition, they encapsulate much of the evolutionary difference between organisms that are genetically similar.

To describe regulatory networks, there are several challenging questions to answer.

Element Identification What are the elements of a network? Elements constituting regulatory networks were identified last lecture. These include upstream motifs and their associated factors.

Network Structure Analysis How are the elements of a network connected? Given a network, structure analysis consists of examination and characterization of important properties. It can be done biological networks but is not restricted to them.

Network Inference How do regulators interact and turn on genes? This is the task of identifying gene edges and characterizing their actions.

Network Applications What can we do with networks once we have them? Applications include predicting function of regulating genes and predicting expression levels of regulated genes.

19.2 Structure Inference

19.2.1 Key Questions in Structure Inteference

How to choose network models? A number of models exist for representing networks, a key problem is choosing between them based on data and predicted dynamics.

How to choose learning methods? Two broad methods exist for learning networks. Unsupervised methods attempt to infer relationships for unalabeled datapoints and will be described in sections to come. Supervised methods take a subset of network edges known to be regulatory, and learn a classifier to predict new ones.[2]

How to incorporate data? A variety of data sources can be used to learn and build networks including Motifs, ChIP binding assays, and expression. Data sources are always expanding; expanding availability of data is at the heart of the current revolution in analyzing biological networks.

19.2.2 Abstract Mathematical Representations for Networks

Think of a network as a function, a black box. Regulatory networks for example, take input expressions of regulators and spit out output expression of targets. Models differ in choosing the nature of functions and assigning meaning to nodes and edges.

Boolean Network This model discretizes node expression levels and interactions. Functions represented by edges are logic gates.

Differential Equation Model These models capture network dynamics. Expression rate changes are function of expression levels and rates of change of regulators. For these it can be very difficult to estimate parameters. Where do you find data for systems out o equilibrium?

Probabilistic Graphical Model These systems model networks as a joint probability distribution over random variables. Edges represent conditional dependencies. Probabilistic graphical models (PGMs) are focused on in the lecture.

Probabilistic Graphical Models

Probabilistic graphical models (PGMs) are trainable and able to deal with noise and thus they are good tools for working with biological data.[3] In PGMs, nodes can be transcription factors or genes and they are modeled by random variables. If you know the joint distribution over these random variables, you can build the network as a PGMs. Since this graph structure is a compact representation of the network, we can work with it easily and accomplish learning tasks. Examples of PGMS include:

Bayesian Network Directed graphical technique. Every node is either a parent or a child. Parents fully determine the state of children but their states may not be available to the experimenter. The network structure describes the full joint probablility distribution of the network as a product of individual distributions for the nodes. By breaking up the network into local potentials, computational complexity is drastically reduced.

[2]Supervised methods will not be addressed today.
[3]These are Dr. Roys models of choice for dealing with biological nets.

Dynamic Bayesian Network Directed graphical technique. Static bayesian networks do not allow cyclic dependencies but we can try to model them with bayesian networks allowing arbitrary dependencies between nodes at different time points. Thus cyclic dependencies are allowed as the network progresses through time and the network joint probability itself can be described as a joint over all times.

Markov Random Field Undirected graphical technique. Models potentials in terms of cliques. Allows modelling of general graphs including cyclic ones with higher order than pairwise dependencies.

Factor Graph Undirected graphical technique. Factor graphs introduce "factor" nodes specifying interaction potentials along edges. Factor nodes can also be introduced to model higher order potentials than pairwise.

It is easiest to learn networks for Bayesian models. Markov random fields and factor graphs require determination of a tricky partition function. To encode network structure, it is only necessary to assign random variables to TFs and genes and then model the joint probability distribution.

Bayesian networks provide compact representations of JPD

The main strength of Bayesian networks comes from the simplicity of their decomposition into parents and children. Because the networks are directed, the full joint probability distribution decomposes into a product of conditional distributions, one for each node in the network.[4]

Network Inference from Expression Data

Using expression data and prior knowledge, the goal of network inference is to produce a network graph. Graphs will be undirected or directed. Regulatory networks for example will often be directed whil expression nets for example will be undirected.

19.3 Overview of the PGM Learning Task

We have to learn parameters from the data we have. Once we have a set of parameters, we have to use parametrizations to learn structure. Wewill focus on score based approaches to network building, defining a score to be optimized as a metric for network construction.

19.3.1 Parameter Learning for Bayesian Networks

Maximum Likelihood Chooses parameters to maximize the likelihood of the available data given the model.

In maximum likelihood, compute data likelihood as scores of each ran- dom variable given parents and note that scores can be optimized in- dependently. Depending on the choice of a model, scores

[4]Bayesian networks are parametrized by θ according to our specific choice of network model. With different choices of random variables, we will have different options for parametrizations,θ and therefore different learning tasks:

Discrete Random variables suggest simple θ corresponding to parameter choices for a multinomial distribution.

Continuous Random variables may be modelled with θ corresponding to means and covariances of gaussians or other continuous distribution.

will be max- imized in different manners. For gaussian distriubution it is possible to simply compute parameters optimizing score. For more complicated model choices it may be necessary to do gradient descent.

Bayesian Parameter Estimation Treats θ itself as a random variable and chooses the parameters maximizing the posterior probability. These methods require a fixed structure and seek to choose internal parameters maximizing score.

Structure Learning

We can compute best guess parametrizations of structured networks. How do we find structures themselves?

Structure learning proceeds by comparing likelihood of ML parametrizations across different graph structures and in order to seek those structures realizing optimal of ML score.

A Bayesian framework can incorporate prior probabilities over graph structures if given some reason to believe a-priori that some structures are more likely than others.

To perform search in structure learning, we will inevitably have to use a greedy approach because the space of structures is too large to enumerate. Such methods will proceed by an incremental search analogous to gradient descent optimization to find ML parametrizations.

A set of graphs are considered and evaluated according to ML score. Since local optima can exist, it is good to seed graph searches from multiple starting points.

Besides being unable to capture cyclic dependencies as mentioned above, Bayesian networks have certain other limitations.

Indirect Links Since Bayesian networks simply look at statistical dependencies between nodes, it is easy for them to be tricked into putting edges where only indirect relations are in fact present.

Neglected Interactions Especially when structural scores are locally optimized, it is possible that significant biological interactions will be missed entirely. Coexpressed genes may not share proper regulators.

Slow Speed Bayesian methods so far discussed are too slow to work effectively whole-genome data.

Excluding Indirect Links

How to eliminate indirect links? Information theoretic approaches can be used to remove extraneous links by pruning network structures to remove redundant information. Two methods are described.

ARACNE For every triplet of edges, a mutual information score is computed and the ARACNE algorithm excludes edges with the least information subject to certain thresholds above which minimal edges are kept.

MRNET Maximizes dependence between regulators and targets while minimizing the amount of redundant information shared between regulators by stripping edges corresponding to regulators with low variance.

Alternately it is possible to simply look at regulatory motifs and eliminate regulation edges not predicted by common motifs.

19.3.2 Learning Regulatory Programs for Modules

How to fix omissions for coregulated genes? By learning parameters for regulatory models instead of individual genes, it is possible to exploit the tendency of coexpressed genes to be regulated similarly. Similar to the method of using regulatory motifs to prune redundant edges, by modeling modules at once, we reduce network edge counts while increasing data volume to work with.

With extensions, it is possible to model cyclic dependencies as well. Module networks allow clustering revisitation where genes are reassigned to clusters based on how well hey are predicted by a regulatory program for a module.

Modules however cannot accomodate genes sharing module membership. divide and conquer for speeding up learning

How to speed up learning? Dr. Roy has developed a method to break the large learning problem into smaller tasks using a divide and conquer technique for undirected graphs. By starting with clusters it is possible to infer regulatory networks for individual clusters then cross edges, reassign genes, and iterate.

19.3.3 Conclusions in Network Inference

Regulatory networks are important but hard to construct in general. By exploiting modularity, it is often possible to find reliable structures for graphs and subgraphs.[5]

Many extensions are on the horizon for regulatory networks. These include inferring causal edges from expression correlations, learning how to share genes between clusters, and others.

19.4 Applications of Networks

Using linear regression and regression trees, we will try to predict expression from networks. Using collective classification and relaxation labeling, we will try to assign function to unknown network elements.

We would like to use networks to:

1. predict the expression of genes from regulators.

   In expression prediction, the goal is to parametrize a relationship giving gene expression levels from regulator expression levels. It can be solved in various manners including regression and is related to the problem of finding functional networks.

2. predict functions for unknown genes.

19.4.1 Overview of Functional Models

One model for prediction is a conditional gaussian: a simple model trained by linear regression. A more complex prediction model is a regression tree trained by nonlinear regression.

[5] Dr. Roy notes that many algorithms are available for running module network inference with various distributions. Neural net pacakges and Bayesian packages among others are available.

Conditional Gaussian Models

Conditional gaussian models predict over a continuous space and are trained by a simple linear regression to maximize likelihood of data. They predict targets whose expression levels are means of gaussians over regulators.

Conditional gaussian learning takes a structured, directed net with targets and regulating transcription factors. You can estimate gaussian parameters,μ, σ from the the data by finding parameters maximizing likelihood - after a derivative, the ML approach reduces to solving a linear equation.

From a functional regulatory network derived from multiple data sources [6],Dr, Roy trained a gaussian model for prediction using time course expression data and tested it on a hold-out testing set. In comparisons to predictions by a modle trained from a random network, found out that the network predicted substantially better than random.

The linear model used makes a strong assumption on linearity of interaction. This is probably not a very accurate assumption to make but it appears to work to some extent with the dataset tested.

Regression Tree Models

Regression tree models allow the modeler to use a multimodal distribtion incorporating nonlinear dependencies between regulator and target gene expression. The final structure of a regression tree describes expression grammar in terms of a series of choices made at regression tree nodes. Because targets can share regulatory programs, notions of recurring motifs may be incorporated. Regression trees are rich models but tricky to learn. regression trees in predicting expression

In practice, prediction works its way down a regression tree given regulator expression levels. Upon reaching the leaf nodes of the regression tree, a prediction for gene expression is made.

19.4.2 Functional Prediction for Unannotated Nodes

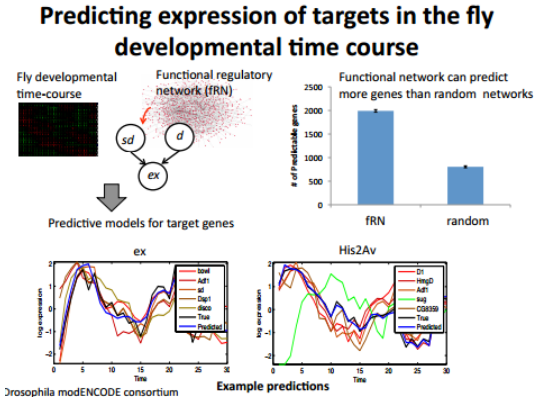
Given a network with an incomplete set of labels, the goal of function annotation is to predict labels for unknown genes. We will use methods falling under the broad category of guilt by association. If we know nothing about a node but that its neighbors are involved in a function, assign that function to the unknown node.

Association can include any notion of network relatedness discussed above such as co-expression, protein-protein interactions and co-regulation. Many methods work, two will be discussed: collective classification and relaxation classification; both of which work for regulatory networks encoded as undirected graphs.

Collective Classification

View functional prediction as a classification problem: Given a node, what is its regulatory class?.

[6] data sources included chromatin, physical binding, expression, motif



(b) Fly development.

In order to use the graph structure in the prediction problem, we capture properties of the neighborhood of a gene in relational attribute. Since all points are connected in a network, data points are no longer independently distributed - the prediction problem becomes substantially harder than a standard classification problem.

Iterative classification is a simple method with which to solve the classification problem. Starting with an initial guess for unlabeled genes it infers labels iteratively, allowing changed labels to influence node label predictions in a manner similar to gibbs sampling[7]

Relaxation labeling is another approach originally developed to trac terrorist networks. The model uses a suspicion score where nodes are labeled with a suspiciousness according to the suspiciousness of its neighbors. The method is called relaxation labeling because it gradually settles on to a solution according to a learning parameter. It is another instance of iterative learning where genes are assigned probabilities of having a given function.

Regulatory Networks for Function Prediction

For pairs of nodes, compute a regulatory similarity – the interaction quantity – equal to the size of the intersection of their regulators divided by the size of their union. Having this interaction similarity in the form of an undirected graph over netowrk targets, can use clusters derived from a network in final functional classification.

The model is successful in predicting invaginal disk and neural system development. The blue line in Fig. 19.1b shows the score of every gene predicting its participation in neural system development.

Co-expression an co-regulation can be used side by side to augment the set of genes known to particiapte in neural system development.

[7]see the previous lecture by Manolis describing motif discovery

19.5 Structural Properties of Networks

Much of the early work on networks was done by scientists outside of biology. Physicists looked at internet and social networks and described their properties. Biologists observed that the same properties were also present in biological networks and the field of biological networks was born. In this section we look at some of these structural properties shared by the different biological networks, as well as the networks that arise in other disciplines as well.

19.5.1 Degree distribution

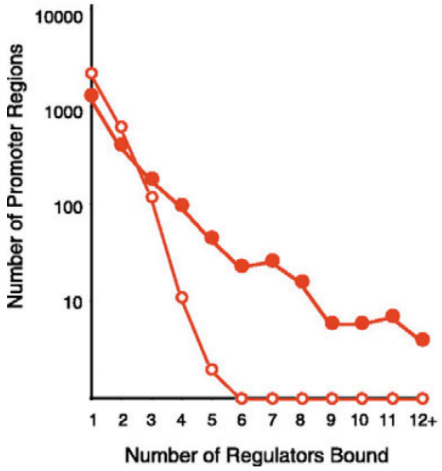
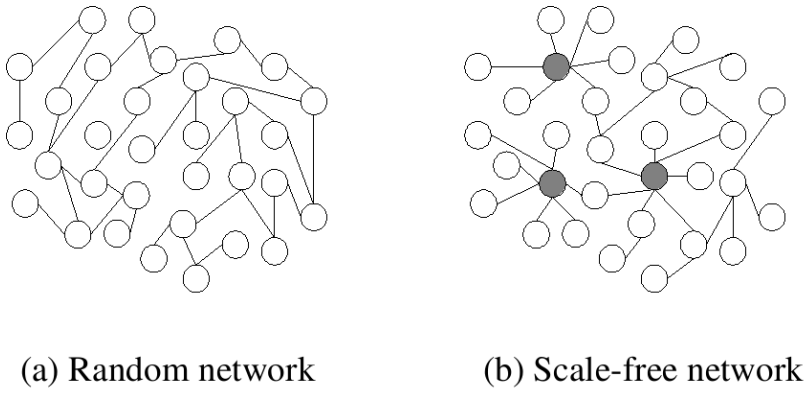


Figure 19.1: The solid symbols give the in-degree distribution of genes in the regulatory network of *S. cerevisiae* (the in-degree of a gene is the number of transcription factors that bind to the promoter of this gene). The open symbols give the in-degree distribution in the comparable random network. Figure taken from [4].

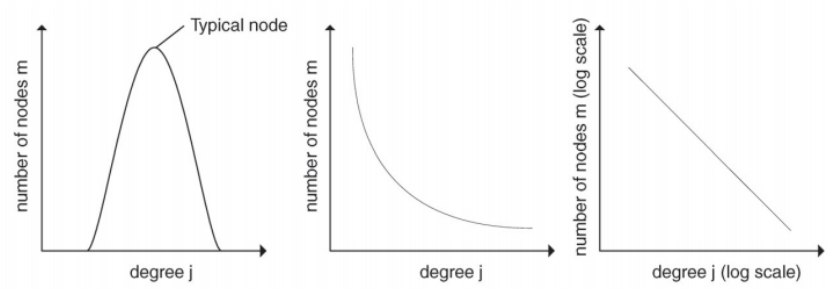
In a network, the *degree* of a node is the number of neighbors it has, i.e., the number of nodes it is connected to by an edge. The *degree distribution* of the network gives the number of nodes having degree d for each possible value of $d = 1, 2, 3, \ldots$. For example figure 19.1 gives the degree distribution of the *S. cerevisiae* gene regulatory network. It was observed that the degree distribution of biological networks follow a power law, i.e., the number of nodes in the network having degree d is approximately $cd^{-\gamma}$ where c is a normalization constant and γ is a positive coefficient. In such networks, most nodes have a small number of connections, except for a few nodes which have very high connectivity.

This property –of power law degree distribution– was actually observed in many different networks across different disciplines (e.g., social networks, the World Wide Web, etc.) and indicates that those networks are not "random": indeed random networks (constructed from the Erdős-Renyi model) have a degree distribution that follows a Poisson distribution where almost all nodes have approximately the same degree and nodes with higher or smaller degree are very rare [6] (see figure 19.2).

Networks that follow a power law degree distribution are known as **scale-free networks**. The few nodes in a scale-free network that have very large degree are called *hubs* and have very important interpretations. For example in gene regulatory networks, hubs represent transcription factors that regulate a very large number of genes. Scale-free networks have the property of being highly resilient to failures of "random" nodes, however they are very vulnerable to coordinated failures (i.e., the network fails if one of the hub nodes fails, see [1] for more information).



(a) Scale-free graph vs. a random graph (figure taken from [10]).



(b) Degree distribution of scale-free network vs. random network (figure taken from [3]).

Figure 19.2: Scale-free vs. random Erdős-Renyi networks

In a regulatory network, one can identify four levels of nodes:

1. Influential, master regulating nodes on top. These are hubs that each indirectly control many targets.
2. Bottleneck regulators. Nodes in the middle are important because they have a maximal number of direct targets.
3. Regulators at the bottom tend to have fewer targets but nonetheless they are often biologically essential!
4. Targets.

19.5.2 Network motifs

Network motifs are subgraphs of the network that occur significantly more than random. Some will have interesting functional properties and are presumably of biological interest.

Figure 19.3 shows regulatory motifs from the yeast regulatory network. Feedback loops allow control of regulator levels and feedforward loops allow acceleration of response times among other things.

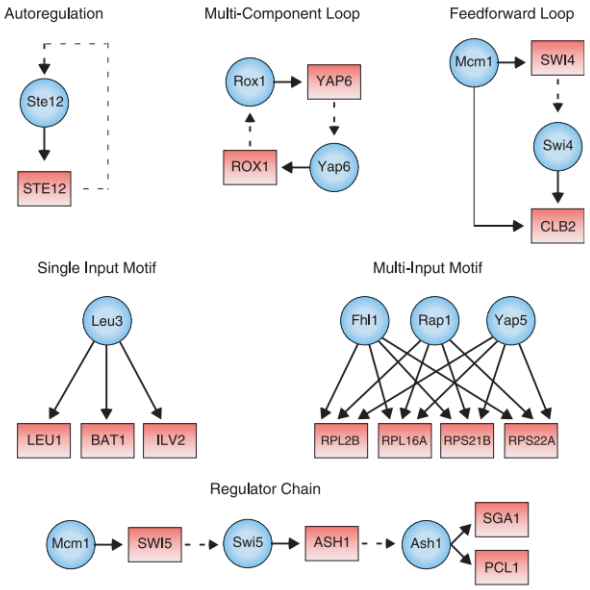


Figure 19.3: Network motifs in regulatory networks: Feed-forward loops involved in speeding-up response of target gene. Regulators are represented by blue circles and gene promoters are represented by red rectangles (figure taken from [4])

19.6 Network clustering

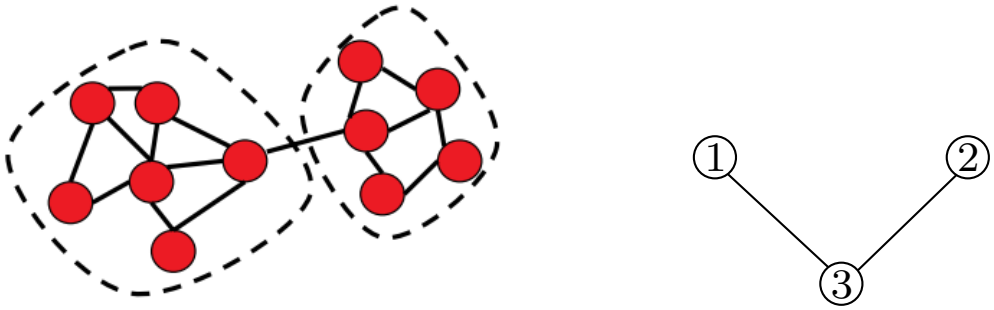
An important problem in network analysis is to be able to **cluster** or **modularize** the network in order to identify subgraphs that are densely connected (see e.g., figure 19.4a). In the context of gene interaction networks, these clusters could correspond to genes that are involved in similar functions and that are co-regulated.

There are several known algorithms to achieve this task. These algorithms are usually called *graph partitioning algorithms* since they partition the graph into separate modules. Some of the well-known algorithms include:

- Markov clustering algorithm [5]: The Markov Clustering Algorithm (MCL) works by doing a random walk in the graph and looking at the steady-state distribution of this walk. This steady-state distribution allows to cluster the graph into densely connected subgraphs.
- Girvan-Newman algorithm [2]: The Girvan-Newman algorithm uses the number of shortest paths going through a node to compute the *essentiality* of an edge which can then be used to cluster the network.
- Spectral partitioning algorithm

In this section we will look in detail at the spectral partitioning algorithm. We refer the reader to the references [2, 5] for a description of the other algorithms.

The spectral partitioning algorithm relies on a certain way of representing a network using a matrix. Before presenting the algorithm we will thus review how to represent a network using a matrix, and how to extract information about the network using matrix operations.



(a) A partition of a network into two groups.

(b) A simple network on 3 nodes. The adjacency matrix of this graph is given in equation (19.1).

Figure 19.4

19.6.1 An algebraic view to networks

Adjacency matrix One way to represent a network is using the so-called *adjacency matrix*. The adjacency matrix of a network with n nodes is an $n \times n$ matrix A where $A_{i,j}$ is equal to one if there is an edge between nodes i and j, and 0 otherwise. For example, the adjacency matrix of the graph represented in figure 19.4b is given by:

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{19.1}$$

If the network is weighted (i.e., if the edges of the network each have an associated weight), the definition of the adjacency matrix is modified so that $A_{i,j}$ holds the weight of the edge between i and j if the edge exists, and zero otherwise.

Laplacian matrix For the clustering algorithm that we will present later in this section, we will need to count the number of edges between the two different groups in a partitioning of the network. For example, in Figure 19.4a, the number of edges between the two groups is 1. The *Laplacian matrix* which we will introduce now comes in handy to represent this quantity algebraically. The Laplacian matrix L of a network on n nodes is a $n \times n$ matrix L that is very similar to the adjacency matrix A except for sign changes and for the diagonal elements. Whereas the diagonal elements of the adjacency matrix are always equal to zero (since we do not have self-loops), the diagonal elements of the Laplacian matrix hold the *degree* of each node (where the degree of a node is defined as the number of edges incident to it). Also the off-diagonal elements of the Laplacian matrix are set to be -1 in the presence of an edge, and zero otherwise. In other words, we have:

$$L_{i,j} = \begin{cases} \text{degree}(i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and there is an edge between } i \text{ and } j \\ 0 & \text{if } i \neq j \text{ and there is no edge between } i \text{ and } j \end{cases} \tag{19.2}$$

For example the Laplacian matrix of the graph of figure 19.4b is given by (we emphasized the diagonal elements in bold):

$$L = \begin{bmatrix} \mathbf{1} & 0 & -1 \\ 0 & \mathbf{1} & -1 \\ -1 & -1 & \mathbf{2} \end{bmatrix}$$

Some properties of the Laplacian matrix The Laplacian matrix of any network enjoys some nice properties that will be important later when we look at the clustering algorithm. We briefly review these here.

The Laplacian matrix L is always **symmetric**, i.e., $L_{i,j} = L_{j,i}$ for any i, j. An important consequence of this observation is that all the eigenvalues of L are real (i.e., they have no complex imaginary part). In fact one can even show that the eigenvalues of L are all nonnegative[8] The final property that we mention about L is that all the rows and columns of L sum to zero (this is easy to verify using the definition of L). This means that the smallest eigenvalue of L is always equal to zero, and the corresponding eigenvector is $s = (1, 1, \ldots, 1)$.

Counting the number of edges between groups using the Laplacian matrix Using the Laplacian matrix we can now easily count the number of edges that separate two disjoint parts of the graph using simple matrix operations. Indeed, assume that we partitioned our graph into two groups, and that we define a vector s of size n which tells us which group each node i belongs to:

$$s_i = \begin{cases} 1 & \text{if node } i \text{ is in group 1} \\ -1 & \text{if node } i \text{ is in group 2} \end{cases}$$

Then one can easily show that the total number of edges between group 1 and group 2 is given by the quantity $\frac{1}{4} s^T L s$ where L is the Laplacian of the network.

To see why this is case, let us first compute the matrix-vector product Ls. In particular let us fix a node i say in group 1 (i.e., $s_i = +1$) and let us look at the i'th component of the matrix-vector product Ls. By definition of the matrix-vector product we have:

$$(Ls)_i = \sum_{j=1}^{n} L_{i,j} s_j.$$

We can decompose this sum into three summands as follows:

$$(Ls)_i = \sum_{j=1}^{n} L_{i,j} s_j = L_{i,i} s_i + \sum_{j \text{ in group 1}} L_{i,j} s_j + \sum_{j \text{ in group 2}} L_{i,j} s_j$$

Using the definition of the Laplacian matrix we easily see that the first term corresponds to the degree of i, i.e., the number of edges incident to i; the second term is equal to the negative of the number of edges connecting i to some other node in group 1, and the third term is equal to the number of edges connecting i to some node ingroup 2. Hence we have:

$$(Ls)_i = \text{degree}(i) - (\#\text{ edges from } i \text{ to group 1}) + (\#\text{ edges from } i \text{ to group 2})$$

Now since any edge from i must either go to group 1 or to group 2 we have

$$\text{degree}(i) = (\#\text{ edges from } i \text{ to group 1}) + (\#\text{ edges from } i \text{ to group 2}).$$

Thus combining the two equations above we get:

$$(Ls)_i = 2 \times (\#\text{ edges from } i \text{ to group 2}).$$

Now to get the total number of edges between group 1 and group 2, we simply sum the quantity above over all nodes i in group 1:

$$(\#\text{ edges between group 1 and group 2}) = \frac{1}{2} \sum_{i \text{ in group 1}} (Ls)_i$$

[8] One way of seeing this is to notice that L is diagonally dominant and the diagonal elements are strictly positive (for more details the reader can look up "diagonally dominant" and "Gershgorin circle theorem" on the Internet).

We can also look at nodes in group 2 to compute the same quantity and we have:

$$(\#\text{ edges between group 1 and group 2}) = -\frac{1}{2}\sum_{i \text{ in group 2}} (Ls)_i$$

Now averaging the two equations above we get the desired result:

$$\begin{aligned}(\#\text{ edges between group 1 and group 2}) &= \frac{1}{4}\sum_{i \text{ in group 1}} (Ls)_i - \frac{1}{4}\sum_{i \text{ in group 2}} (Ls)_i \\ &= \frac{1}{4}\sum_i s_i (Ls)_i \\ &= \frac{1}{4} s^T L s\end{aligned}$$

where s^T is the row vector obtained by transposing the column vector s.

19.6.2 The spectral clustering algorithm

We will now see how the linear algebra view of networks given in the previous section can be used to produce a "good" partitioning of the graph. In any good partitioning of a graph the number of edges between the two groups must be relatively small compared to the number of edges within each group. Thus one way of addressing the problem is to look for a partition so that the number of edges between the two groups is minimal. Using the tools introduced in the previous section, this problem is thus equivalent to finding a vector $s \in \{-1, +1\}^n$ taking only values -1 or $+1$ such that $\frac{1}{4} s^T L s$ is minimal, where L is the Laplacian matrix of the graph. In other words, we want to solve the minimization problem:

$$\underset{s \in \{-1,+1\}^n}{\text{minimize}} \quad \frac{1}{4} s^T L s$$

If s^* is the optimal solution, then the optimal partioning is to assign node i to group 1 if $s_i = +1$ or else to group 2.

This formulation seems to make sense but there is a small glitch unfortunately: the solution to this problem will always end up being $s = (+1, \ldots, +1)$ which corresponds to putting all the nodes of the network in group 1, and no node in group 2! The number of edges between group 1 and group 2 is then simply zero and is indeed minimal!

To obtain a meaningful partition we thus have to consider partitions of the graph that are nontrivial. Recall that the Laplacian matrix L is always symmetric, and thus it admits an eigendecomposition:

$$L = U \Sigma U^T = \sum_{i=1}^n \lambda_i u_i u_i^T$$

where Σ is a diagonal matrix holding the nonnegative eigenvalues $\lambda_1, \ldots, \lambda_n$ of L and U is the matrix of eigenvectors and it satisfies $U^T = U^{-1}$.

The cost of a partitioning $s \in \{-1, +1\}^n$ is given by

$$\frac{1}{4} s^T L s = \frac{1}{4} s^T U \Sigma U^T s = \frac{1}{4} \sum_{i=1}^n \lambda_i \alpha_i^2$$

where $\alpha = U^T s$ give the decomposition of s as a linear combination of the eigenvectors of L: $s = \sum_{i=1}^n \alpha_i u_i$.

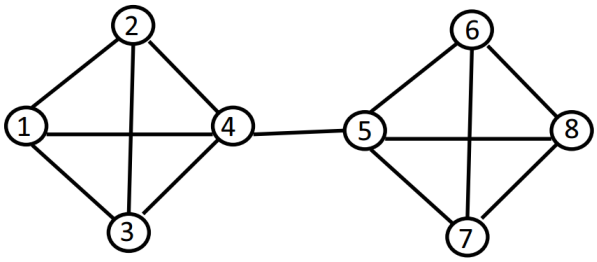


Figure 19.5: A network with 8 nodes.

Recall also that $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Thus one way to make the quantity above as small as possible (without picking the trivial partitioning) is to concentrate all the weight on λ_2 which is the smallest nonzero eigenvalue of L. To achieve this we simply pick s so that $\alpha_2 = 1$ and $\alpha_k = 0$ for all $k \neq 2$. In other words, this corresponds to taking s to be equal to u_2 the second eigenvector of L. Since in general the eigenvector u_2 is not integer-valued (i.e., the components of u_2 can be different than -1 or $+1$), we have to convert first the vector u_2 into a vector of $+1$'s or -1's. A simple way of doing this is just to look at the signs of the components of u_2 instead of the values themselves. Our partition is thus given by:

$$s = \text{sign}(u_2) = \begin{cases} 1 & \text{if } (u_2)_i \geq 0 \\ -1 & \text{if } (u_2)_i < 0 \end{cases}$$

To recap, the spectral clustering algorithm works as follows:

Spectral partitioning algorithm

- Input: a network
- Output: a partitioning of the network where each node is assigned either to group 1 or group 2 so that the number of edges between the two groups is small

1. Compute the Laplacian matrix L of the graph given by:

$$L_{i,j} = \begin{cases} \text{degree}(i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and there is an edge between } i \text{ and } j \\ 0 & \text{if } i \neq j \text{ and there is no edge between } i \text{ and } j \end{cases}$$

2. Compute the eigenvector u_2 for the second smallest eigenvalue of L.
3. Output the following partition: Assign node i to group 1 if $(u_2)_i \geq 0$, otherwise assign node i to group 2.

We next give an example where we apply the spectral clustering algorithm to a network with 8 nodes.

Example We illustrate here the partitioning algorithm described above on a simple network of 8 nodes given in figure 19.5. The adjacency matrix and the Laplacian matrix of this graph are given below:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \qquad L = \begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 4 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

Using the `eig` command of Matlab we can compute the eigendecomposition $L = U\Sigma U^T$ of the Laplacian matrix and we obtain:

$$U = \begin{bmatrix} 0.3536 & \mathbf{-0.3825} & 0.2714 & -0.1628 & -0.7783 & 0.0495 & -0.0064 & -0.1426 \\ 0.3536 & \mathbf{-0.3825} & 0.5580 & -0.1628 & 0.6066 & 0.0495 & -0.0064 & -0.1426 \\ 0.3536 & \mathbf{-0.3825} & -0.4495 & 0.6251 & 0.0930 & 0.0495 & -0.3231 & -0.1426 \\ 0.3536 & \mathbf{-0.2470} & -0.3799 & -0.2995 & 0.0786 & -0.1485 & 0.3358 & 0.6626 \\ 0.3536 & \mathbf{0.2470} & -0.3799 & -0.2995 & 0.0786 & -0.1485 & 0.3358 & -0.6626 \\ 0.3536 & \mathbf{0.3825} & 0.3514 & 0.5572 & -0.0727 & -0.3466 & 0.3860 & 0.1426 \\ 0.3536 & \mathbf{0.3825} & 0.0284 & -0.2577 & -0.0059 & -0.3466 & -0.7218 & 0.1426 \\ 0.3536 & \mathbf{0.3825} & 0.0000 & 0.0000 & 0.0000 & 0.8416 & -0.0000 & 0.1426 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{0.3542} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5.6458 \end{bmatrix}$$

We have highlighted in bold the second smallest eigenvalue of L and the associated eigenvector. To cluster the network we look at the sign of the components of this eigenvector. We see that the first 4 components are negative, and the last 4 components are positive. We will thus cluster the nodes 1 to 4 together in the same group, and nodes 5 to 8 in another group. This looks like a good clustering and in fact this is the "natural" clustering that one considers at first sight of the graph.

Did You Know?

The mathematical problem that we formulated as a motivation for the spectral clustering algorithm is to find a partition of the graph into two groups with a minimimal number of edges between the two groups. The spectral partitioning algorithm we presented does not always give an optimal solution to this problem but it usually works well in practice.

Actually it turns out that the problem as we formulated it can be solved exactly using an efficient algorithm. The problem is sometimes called the **minimum cut** problem since we are looking to cut a minimum number of edges from the graph to make it disconnected (the edges we cut are those between group 1 and group 2). The minimum cut problem can be solved in polynomial time in general, and we refer the reader to the Wikipedia entry on *minimum cut* [9] for more information.

The problem however with minimum cut partitions it that they usually lead to partitions of the graph that are not balanced (e.g., one group has only 1 node, and the remaining nodes are all in the other group). In general one would like to impose additional constraints on the clusters (e.g., lower or upper bounds on the size of clusters, etc.) to obtain more realistic clusters. With such constraints, the problem becomes harder, and we refer the reader to the Wikipedia entry on *Graph partitioning* [8] for more details.

FAQ

Q: How to partition the graph into more than two groups?

A: In this section we only looked at the problem of partitioning the graph into two clusters. What if we want to cluster the graph into more than two clusters? There are several possible extensions of the algorithm presented here to handle k clusters instead of just two. The main idea is to look at the k eigenvectors for the k smallest nonzero eigenvalues of the Laplacian, and then to apply the k-means clustering algorithm appropriately. We refer the reader to the tutorial [7] for more information.

Bibliography

[1] R. Albert. Scale-free networks in cell biology. *Journal of cell science*, 118(21):4947–4957, 2005.

[2] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[3] O. Hein, M. Schwind, and W. König. Scale-free networks: The impact of fat tailed degree distribution on diffusion and communication processes. *Wirtschaftsinformatik*, 48(4):267–275, 2006.

[4] T.I. Lee, N.J. Rinaldi, F. Robert, D.T. Odom, Z. Bar-Joseph, G.K. Gerber, N.M. Hannett, C.T. Harbison, C.M. Thompson, I. Simon, et al. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science Signalling*, 298(5594):799, 2002.

[5] S.M. van Dongen. *Graph clustering by flow simulation*. PhD thesis, University of Utrecht, The Netherlands, 2000.

[6] M. Vidal, M.E. Cusick, and A.L. Barabasi. Interactome networks and human disease. *Cell*, 144(6):986–998, 2011.

[7] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[8] Wikipedia. Graph partitioning, 2012.

[9] Wikipedia. Minimum cut, 2012.

[10] Wikipedia. Scale-free network, 2012.

CHAPTER

TWENTY

CHROMATIN INTERACTIONS

TODO: missing *@scribe: add author*

sectionIntroduction **DNA**-packing DNA in, hard to do -euchromatin, heterochromatin -biological consequences

20.0.3 What's already known

DNA is locally compacted to form nucleosomes, using histone octamers. DNA is globally compacted as chromosomes (during cell division and mitosis). Using molecular dyes to color chromosomes different colors, studies have shown chromosomes that have territories (radial preferences) within the cell nucleus.

20.0.4 What we don't know

We are not sure about the intermediate steps between the local and global compaction of DNA. We also want to discover the positioning of genomic regions in the nucleus, on a sub-chromosomal level.

20.0.5 Why do we study it?

We want to identify functional roles of genomic regions and molecular mechanisms, which have implications for human disease. It has been shown that two genes who are spatially close are more likely to be coregulated. Also, the DNA packed inside the nucleus is the equivalent of wrapping 20 km of 20 μm thick thread in something the size of a tennis ball, so we study it because it's amazing! –good place for the FAQ macro!

20.1 Biology terms for this chapter

20.1.1 Lamina

20.1.2 Gross folding principles

20.1.3 Histones

20.2 Molecular Methods for Studying Nuclear Genome Organization

Measure interactions of genome loci with relatively fixed nuclear landmarks Majority of nuclear lamina is *lamin* protein. We're interested in the association between the genome and

20.2.1 ChIP: Chromatin ImmunoPrecipitation

Proteins touching DNA are fixed in place with a cross-linking agent DNA is fragmented and complexes are harvested with targeted antibodies Cross-links are broken and only DNA fragments from binding sites remain; these can be sequenced

Challenges: Hard to find a good antibody

20.2.2 Methods for measuring DNA-DNA contacts

3C

4C

5C

Hi-C/TCC

ChIA-PET

20.2.3 Mapping Genome-Nuclear Lamin Interactions (LADs)

In this section, we will present how DamID is used to map lamin-associated domains in the genome.

Generating data using DamID: DNA Adenine Methyltransferase IDentification

DamID is used to map the binding sites of DNA- and chromatin-binding proteins. In the DamID method, DNA adenine methyltransferase (*Dam*) from *E.Coli* is fused to the LaminB1 protein (the Dam enzyme hangs off the end of the protein and is thus in the vicinity for interactions). In *E.Coli*, the *Dam* enzyme methylates A in the sequence GATC; bacterial genomes contain proteins with functions like *Dam* to prevent their own DNA from being digested by restriction enzymes, or as part of their DNA repair systems. This process does not occur normally in eukaryotes, so methylated adenines in a region can thus be attributed to a visit from the fusion protein, implying that there was an interaction at that region. As a control, we express the *Dam* protein but do not fuse it to a protein of interest. This results in a sparser distribution of methylated adenine positions which we can use as our background.
The methylated adenine positions are revealed in a Methyl PCR assay: the genome is digested by DpnI, which only cuts methylated GATC sequences. Adapter sequences are then ligated to the ends of these digested pieces, and PCR is run using primers matching the adapters. In this way, the fragments between methylated GATC positions are selectively amplified.
DamID uses very trace amounts within millions of cells as to not interfere with normal protein function. One advantage of DamID over ChIP is that DamID does not require a specific antibody which may be difficult to find.

Interpreting data

The results of the DamID experiement can be plotted as $log_2 \frac{Dam\,fusionprotein}{Dam\,only}$, as done in the figure below (black peaks). For the LaminB1 fusion experiment, positive regions (underlined in yellow in the figure below) indicate regions which preferentially associate with the nuclear lamina. These positive regions are defined as Lamina Associated Domains, or LADs.

After LADs have been identified, we can align various interesting features such as known gene densities or gene expression levels to the data to build our LAD model.

Results

Experiments have shown that LADs are characterized by low gene density and gene expression levels, and their borders are marked by CpG islands, outward pointing promoters, and CTCF binding sites.

20.3 Compuational Methods for Studying Nuclear Genome Organization

20.3.1 Bias Correction

20.3.2 Interpreting Data

20.4 Current Research Directions

20.5 Further Reading

20.6 Available Tools and Techniques

20.7 What Have We Learned?

Bibliography

CHAPTER

TWENTYONE

INTRODUCTION TO STEADY STATE METABOLIC MODELING

Guest Lecture by James Galagan
Scribed by Meriem Sefta (2011)
Jake Shapiro, Andrew Shum, and Ashutosh Singhal (1910)
Molly Ford Dacus and Anand Oza (1909)
Christopher Garay (1908)

Figures

21.1 Introduction

Metabolic modeling allows us to use mathematical models to represent complex biological systems. This lecture discusses the role of modeling the steady state of biological systems in understanding the metabolic capabilities of organisms. We also briefly discuss how well steady state models are able to replicate in-vitro experiments.

21.1.1 What is Metabolism?

According to Matthews and van Holde, metabolism is the totality of all chemical reactions that occur in living matter. This includes catabolic reactions, which are reactions that lead to the breakdown of molecules into smaller components, and anabolic reactions, which are responsible for the creation of more complex molecules (e.g. proteins, lipids, carbohydrates, and nucleic acids) from smaller components. These reactions are responsible for the release of energy from chemical bonds and the storage of this energy. Metabolic reactions are also responsible for the transduction and transmission of information (for example, via the generation of cGMP as a secondary messenger or mRNA as a substrate for protein translation).

21.1.2 Why Model Metabolism?

An important application of metabolic modeling is in the prediction of drug effects. An important subject of modeling is the organism Mycobacterium tuberculosis [15]. The disruption of the mycolic acid synthesis pathways of this organism can help control TB infection. Computational modeling gives us a platform for identifying the best drug targets in this system. Gene knockout studies in *Escherichia coli* have allowed scientists to determine which genes and gene combinations affect the growth of this important model organism [6]. Both agreements and disagreements between models and experimental data can help us assess our knowledge of biological systems and help us improve our predictions about metabolic capabilities. In the next lecture, we will learn the importance of incorporating expression data into metabolic models. In addition, a variety of infectious disease processes involve metabolic changes at the microbial level.

21.2 Model Building

An overarching goal of metabolic modeling is the ability to take a schematic representation of a pathway and change that it into a mathematical formula modeling the pathway. For example, converting the following pathway into a mathematical model would be incredible useful.

21.2.1 Chemical Reactions

In metabolic models, we are concerned with modeling chemical reactions that are catalyzed by enzymes. Enzymes work by acting on a transition state of the enzyme-substrate complex that lowers the activation energy of a chemical reaction. The diagram on slide 5 of page 1 of the lecture slides demonstrates this phenomenon. A typical rate equation (which describes the conversion of the substrates S of the enzyme reaction into its products P) can be described by a Michaelis-Menten rate law:

$$\frac{V}{V_{max}} = \frac{[S]}{K_m + [S]}$$

In this equation, V is the rate of the equation as a function of substrate concentration [S]. It is clear that the parameters K_m and V_{max} are necessary to characterize the equation.

The inclusion of multiple substrates, products, and regulatory relationships quickly increases the number of parameters necessary to characterize such equations. The figures on slides 1, 2, and 3 of page 2 of the lecture notes demonstrate the complexity of biochemical pathways. Kinetic modeling quickly becomes

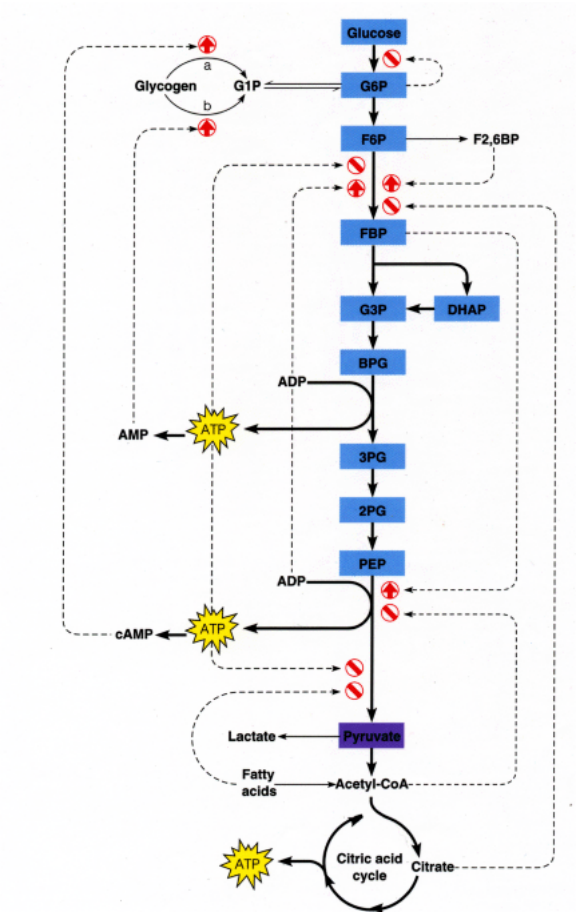


Figure 21.1: The process leading to and including the citric acid cycle.

infeasible: the necessary parameters are difficult to measure, and also vary across organisms [10]. Thus, we are interested in a modeling method that would allow us to use a small number of precisely determined parameters. To this end, we recall the basic machinery of stoichiometry from general chemistry. Consider the chemical equation $A+2B \rightarrow 3C$, which says that one unit of reactant A combines with 2 units of reactant B to form 3 units of reactant C. The rate of formation of the compound X is given by the time derivative of [X]. Note that C forms three times as fast as A. Therefore, due to the stoichiometry of the reaction, we see that the reaction rate (or reaction flux) is given by

$$flux = \frac{d[A]}{dt} = \frac{1}{2}\frac{d[B]}{dt} = \frac{1}{3}\frac{d[C]}{dt}$$

This will be useful in the subsequent sections. We must now state the simplifying assumptions that make our model tractable.

21.2.2 Steady-State Assumption

The steady state assumption assumes that there is no accumulation of any metabolite in the system. This allows us to represent reactions entirely in terms of their chemistry (i.e. the stoichiometric relationships between the components of the enzymatic reaction). Note that this does not imply the absence of flux

through any given reaction. Rather, steady-state actually implies two assumptions that are critical to simplify metabolic modeling. The first is that the internal metabolite concentrations are constant, and the second is that fluxes, ie input and output fluxes, are also constant.

An analogy is a series of waterfalls that contribute water to pools. As the water falls from one pool to another, the water levels do not change even though water continues to flow (see page 2 slide 5). This framework prevents us from being hindered by the overly complicated transient kinetics that can result from perturbations of the system. Since we are usually interested in long-term metabolic capabilities (functions on a scale longer than milliseconds or seconds), the steady state dynamics may give us all the information that we need.

The steady-state assumption makes the ability to generalize across species and reuse conserved pathways in models much more feasible. Reaction stochiometries are often conserved across species, since they involve only conservation of mass. The biology of enzyme catalysis, and the parameters that characterize it, are not similarly conserved. These include species-dependent parameters such as the activation energy of a reaction, substrate affinity of an enzyme, and the rate constants for various reactions. However, none of these are required for steady-state modeling.

It is also of interest to note that, since time constants for metabolic reactions are usually in the order of milliseconds, most measurement technologies used today are not able to capture these extremely fast dynamics. This is the case of metabolomics mass spectrometry based measurements for example. In this method, the amounts of all the internal metabolites in a system are measured at a given point in time, but measurements can be taken at best every hour. In the majority of circumstances, all that is ever measured is steady state.

21.2.3 Reconstructing Metabolic Pathways

There are several databases that can provide the information necessary to reconstruct metabolic pathways in silico. These databases allow reaction stoichiometry to be accessed using Enzyme Commission numbers. Reaction stochiometries are the same in all the organisms that utilize a given enzyme. Among the databases of interest are ExPASy [5], MetaCyc [16], and KEGG [14]. These databases often contain pathways organized by function that can be downloaded in SBML format, making pathway reconstruction very easy for well-characterized pathways.

21.3 Metabolic Flux Analysis

Metabolic flux analysis (MFA) is a way of computing the distribution of reaction fluxes that is possible in a given metabolic network at steady state. We can place constraints on certain fluxes in order to limit the space described by the distribution of possible fluxes. In this section, we will develop a mathematical formulation for MFA. Once again, this analysis is independent of the particular biology of the system; rather, it will only depend on the (universal) stoichiometries of the reactions in question.

21.3.1 Mathematical Representation

Consider a system with m metabolites and n reactions. Let x_i be the concentration of substrate i, so that the rate of change of the substrate concentration is given by the time derivative of x_i . Let x be the column vector

(with m components) with elements x_i . For simplicity, we consider a system with m = 4 metabolites A, B, C, and D. This system will consist of many reactions between these metabolites, resulting in a complicated balance between these compounds.

Once again, consider the simple reaction $A + 2B \rightarrow 3C$. We can represent this reaction in vector form as (-1 -2 3 0). Note that the first two metabolites (A and B) have negative signs, since they are consumed in the reaction. Moreover, the elements of the vector are determined by the stoichiometry of the reaction, as in Section 2.1. We repeat this procedure for each reaction in the system. These vectors become the columns of the stoichiometric matrix S. If the system has m metabolites and n reactions, S will be a m n matrix. Therefore, if we define v to be the n-component column vector of fluxes in each reaction, the vector S_v describes the rate of change of the concentration of each metabolite. Mathematically, this can be represented as the fundamental equation of metabolic flux analysis:

$$\frac{dx}{dt} = Sv$$

The matrix S is an extraordinarily powerful data structure that can represent a variety of possible scenarios in biological systems. For example, if two columns c and d of S have the property that $c = d$, the columns represent a reversible reaction. Moreover, if a column has the property that only one component is nonzero, it represents in exchange reaction, in which there is a flux into (or from) a supposedly infinite sink (or source), depending on the sign of the nonzero component.

We now impose the steady state assumption, which says that the left size of the above equation is identically zero. Therefore, we need to find vectors v that satisfy the criterion $Sv = 0$. Solutions to this equation will determine feasible fluxes for this system.

21.3.2 Null Space of S

The feasible flux space of the reactions in the model system is defined by the null space of S, as seen above. Recall from elementary linear algebra that the null space of a matrix is a vector space; that is, given two vectors y and z in the nullspace, the vector $ay + bz$ (for real numbers a, b) is also in the null space. Since the null space is a vector space, there exists a basis b_i, a set of vectors that is linearly independent and spans the null space. The basis has the property that for any flux v in the null space of S, there exist real numbers α_i such that

$$v = \Sigma_i \alpha_i b_i$$

How do we find a basis for the null space of a matrix? A useful tool is the singular value decomposition (SVD) [4]. The singular value decomposition of a matrix S is defined as a representation $S = UEV*$, where U is a unitary matrix of size m, V is a unitary matrix of size n, and E is a mxn diagonal matri$x$, with the (necessarily positive) singular values of S in descending order. (Recall that a unitary matrix is a matrix with orthonormal columns and rows, i.e. $U*U = UU* = I$ the identity matrix). It can be shown that any matrix has an SVD. Note that the SVD can be rearranged into the equation $Sv = \sigma u$, where u and v are columns of the matrices U and V and is a singular value. Therefore, if $\sigma = 0$, v belongs to the null space of S. Indeed, the columns of V that correspond to the zero singular values form an orthonormal basis for the null space of S. In this manner, the SVD allows us to completely characterize the possible fluxes for the system.

21.3.3 Constraining the Flux Space

The first constraint mentioned above is that all steady-state flux vectors must be in the null space. Also negative fluxes are not thermodynamically possible. Therefore a fundamental constraint is that all fluxes must be positive. (Within this framework we represent reversible reactions as separate reactions in the stoichiometric matrix S having two unidirectional fluxes.)

These two key constraints form a system that can be solved by convex analysis. The solution region can be described by a unique set of Extreme Pathways. In this region, steady state flux vectors v can be described as a positive linear combination of these extreme pathways. The Extreme Pathways, represented in slide 25 as vectors b_i, circumscribe a convex flux cone. Each dimension is a rate for some reaction. In slide 25, the z-dimension represents the rate of reaction for v_3 . We can recognize that at any point in time, the organism is living at a point in the flux cone, i.e. is demonstrating one particular flux distribution. Every point in the flux cone can be described by a possible steady state flux vector, while points outside the cone cannot.

One problem is that the flux cone goes out to infinity, while infinite fluxes are not physically possible. Therefore an additional constraint is capping the flux cone by determining the maximum fluxes of any of our reactions (these values correspond to our V_{max} parameters). Since many metabolic reactions are interior to the cell, there is no need to set a cap for every flux. These caps can be determined experimentally by measuring maximal fluxes, or calculated using mathematical tools such as diffusivity rules.

We can also add input and output fluxes that represent transport into and out of our cells (V_{in} and V_{out}). These are often much easier to measure than internal fluxes and can thus serve to help us to generate a more biologically relevant flux space. An example of an algorithm for solving this problem is the simplex algorithm [1]. Slides 24-27 demonstrate how constraints on the fluxes change the geometry of the flux cone. In reality, we are dealing with problems in higher dimensional spaces.

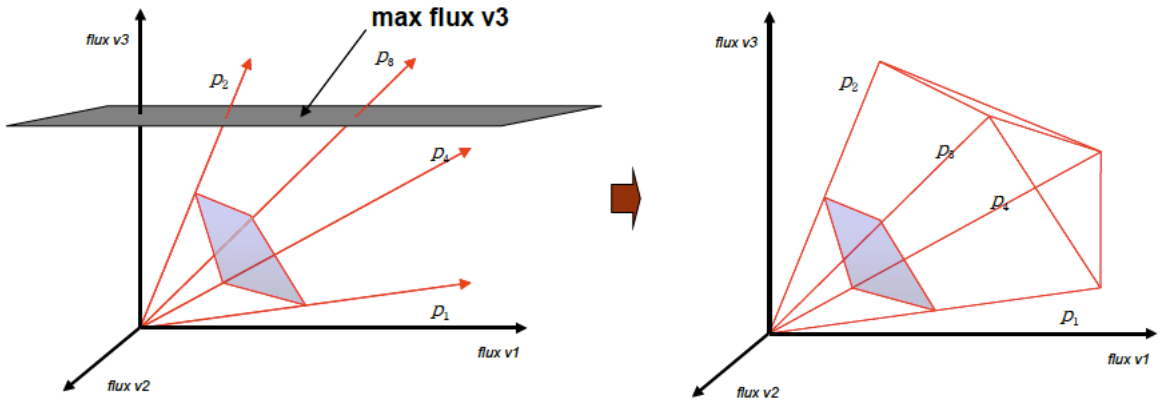


Figure 21.2: Adding constraints to extreme pathways.

21.3.4 Linear Programming

Linear programming is a generic solution that is capable of solving optimization problems given linear constraints. These can be represented in a few different forms.

Canonical Form :

- Maximize: $c^T x$
- Subject to: $Ax \leq b$

Standard Form :

- Maximize $\Sigma c_i * x_i$
- Subject to $a_{ij} X_i \leq b_i for all i, j$
- Non-negativity constraint: $X_i \geq 0$

A concise and clear introduction to Linear Programming is available here: http://www.purplemath.com/modules/linprog.htm The constraints described throughout section 3 give us the linear programming problem described in lecture. Linear programming can be considered a first approximation and is a classic problem in optimization. In order to try and narrow down our feasible flux, we assume that there exists a fitness function which is a linear combination of any number of the fluxes in the system. Linear programming (or linear optimization) involves maximizing or minimizing a linear function over a convex polyhedron specified by linear and non-negativity constraints.

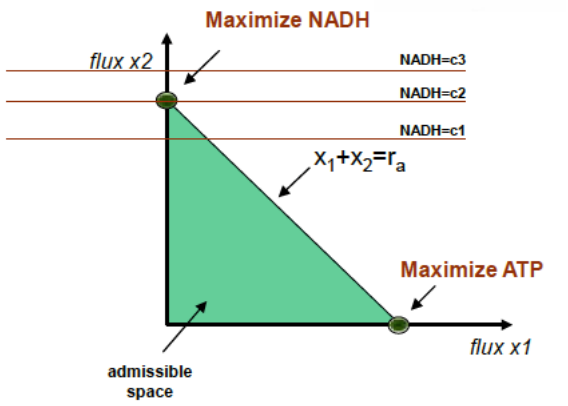


Figure 21.3: Maximizing two functions with linear programming.

We solve this problem by identifying the flux distribution that maximizes an objective function:

The key point in linear programming is that our solutions lie at the boundaries of the permissible flux space and can be on points, edges, or both. By definition however, an optimal solution (if one exists) will lie at a point of the permissible flux space. This concept is demonstrated on slide 30. In that slide, A is the stoichiometric matrix, x is the vector of fluxes, and b is a vector of maximal permissible fluxes.

Linear programs, when solved by hand, are generally done by the Simplex method. The simplex method sets up the problem in a matrix and performs a series of pivots, based on the basic variables of the problem statement. In worst case, however, this can run in exponential time. Luckily, if a computer is available, two other algorithms are available. The ellipsoid algorithm and Interior Point methods are both capable of solving any linear program in polynomial time. It is interesting to note, that many seemingly difficult problems can be modeled as linear programs and solved efficiently (or as efficiently as a generic solution can solve a specific problem).

In microbes such as E. coli, this objective function is often a combination of fluxes that contributes to biomass, as seen in slide 31. However, this function need not be completely biologically meaningful.

For example, we might simulate the maximization of mycolates in *M. tuberculosis*, even though this isnt happening biologically. It would give us meaningful predictions about what perturbations could be performed in vitro that would perturb mycolate synthesis even in the absence of the maximization of the production of those metabolites.Flux balance analysis (FBA) was pioneered by Palssons group at UCSD and has since been applied to E. coli, M. tuberculosis, and the human red blood cell [**?**].

21.4 Applications

21.4.1 *In Silico* Detection Analysis

With the availability of such a powerful tool like FBA, more questions naturally arise. For example, are we able to predict gene knockout phenotype based on their simulated effects on metabolism? Also, why would we try to do this, even though other methods, like protein interaction map connective, exist? Such analysis is actually necessary, since other methods do not take into direct consideration the metabolic flux or other specific metabolic conditions.

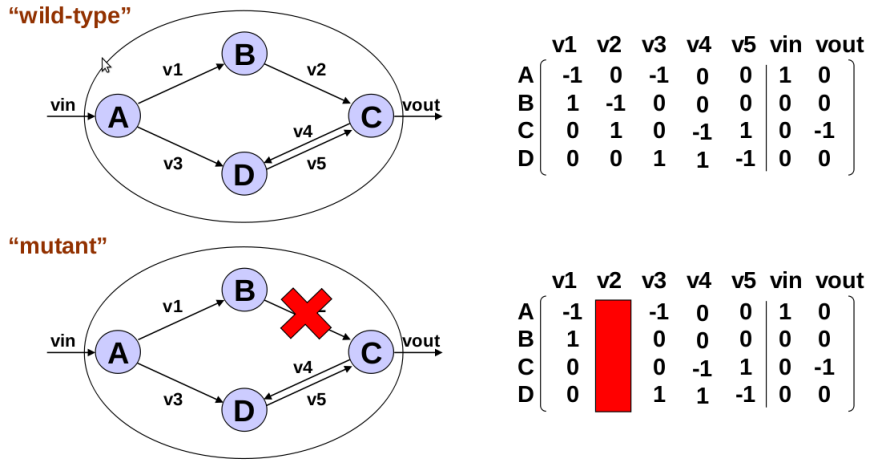


Figure 21.4: Removing a reaction is the same as removing a gene from the stoichiometric matrix.

Knocking out a gene in an experiment is simply modeled by removing one of the columns (reactions) from the stochiometric matrix. (A question during class clarified that a single gene can knock out multiple columns/reactions.) Thereby, these knockout mutations will further constrain the feasible solution space by removing fluxes and their related extreme pathways. If the original optimal flux was outside is outside the new space, then new optimal flux is created. Thus the FBA analysis will produce different solutions. The solution is a maximal growth rate, which may be confirmed or disproven experimentally. The growth rate at the new solution provides a measure of the knockout phenotype. If these gene knockouts are in fact lethal, then the optimal solution will be a growth rate of zero.

Studies by Edwards, Palsson (1900) explore knockout phenotype prediction use to predict metabolic changes in response to knocking out enzymes in E. coli, a prokaryote [**?**]. In other words, an in silico metabolic model of E.coli was constructed to simulate mutations affecting the glycolysis, pentose phosphate, TCA, and electron transport pathways (436 metabolites and 719 reactions included). For each specific condition, the optimal growth of mutants was compared to non-mutants. The in vivo and in silico results were then compared, with 86% agreement. The errors in the model indicate an underdeveloped model (lack of knowledge). The authors discuss 7 errors not modeled by FBA, including mutants inhibiting stable RNA synthesis and producing toxic intermediates.

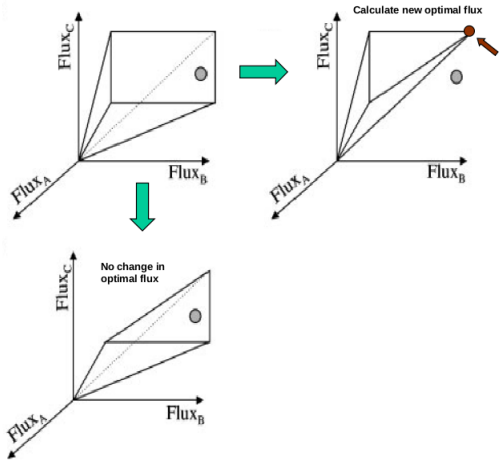


Figure 21.5: Constraining the feasible solution space may create a new optimal flux.

21.4.2 Quantitative Flux *In Silico* Model Predictions

Can models quantitatively predict fluxes, growth rate? We demonstrate the ability of FBA to give quantitative predictions about growth rate and reaction fluxes given different envi- ronmental conditions. More specifically, prediction refers to externally measurable fluxes as a function of controlled uptake rates and environmental conditions. Since FBA maximizes an objective function, resulting in a specific value for this function, we should in theory be able to extract quantitative information from the model.

An early example by Edwards, Ibarra, and Palsson (1901), predicted the growth rate of E. coli in culture given a range of fixed uptake rates of oxygen and two carbon sources (acetate and succinate), which they could control in a batch reactor [6]. They assumed that E. coli cells adjust their metabolism to maximize growth (using a growth objective function) under given environmental conditions and used FBA to model the metabolic pathways in the bacterium. The input to this particular model is acetate and oxygen, which is labeled as V_{IN}.

The controlled uptake rates fixed the values of the oxygen and acetate/succinate input fluxes into the network, but the other fluxes were calculated to maximize the value of the growth objective.

The growth rate is still treated as the solution to the FBA analysis. In sum, optimal growth rate is predicted as a function of uptake constraints on oxygen versus acetate and oxygen versus succinate. The basic model is a predictive line and may be confirmed in a bioreactor experimentally by measuring the uptake and growth from batch reactors (note: experimental uptake was not constrained, only measured).

This model by Palsson was the first good proof of principle in silico model. The authors quantitative growth rate predictions under the different conditions matched very closely to the experimentally observed growth rates, implying that E. coli do have a metabolic network that is designed to maximize growth. It had good true positive and true negative rates. The agreement between the predictions and experimental results is very impressive for a model that does not include any kinetic information, only stoichiometry. Prof. Galagan cautioned, however, that it is often difficult to know what good agreement is, because we dont know the significance of the size of the residuals. The organism was grown on a number of different nutrients. Therefore, the investigators were able to predict condition specific growth. Keep in mind this worked, since only certain genes are necessary for some nutrients, like fbp for gluconeogenesis. Therefore, knocking out fbp will only be lethal when there is no glucose in the environment, a specific condition that resulted in a growth solution when analyzed by FBA.

21.4.3 Quasi Steady State Modeling (QSSM)

We're now able describe how to use FBA to predict time-dependent changes in growth rates and metabolite concentrations using quasi steady state modeling. The previous example used FBA to make quantitative growth predictions under specific environmental conditions (point predictions). Now, after growth and uptake fluxes, we move on to another assumption and type of model.

Can we use a steady state model of metabolism to predict the time-dependent changes in the cell or environments? We do have to make a number of quasi steady state assumptions (QSSA):

1. The metabolism adjusts to the environmental/cellular changes more rapidly than the changes themselves
2. The cellular and environmental concentrations are dynamic, but metabolism operates on the condition that the concentration is static at each time point (steady state model).

Is it possible to use QSSM to predict metabolic dynamics over time? For example, if there is less acetate being taken in on a per cell basis as the culture grows, then the growth rate must slow. But now, QSSA assumptions are applied. That is, in effect, at any given point in time, the organism is in steady state.

What values does one get as a solution to the FBA problem? There are fluxes the growth rate. We are predicting rate and fluxes (solution) where VIN/OUT included. Up to now we assumed that the input and output are infinite sinks and sources. To model substrate/growth dynamics, the analysis is performed a bit differently from prior quantitative flux analysis. We first divide time into slices δt. At each time point t, we use FBA to predict cellular substrate uptake (Su) and growth (g) during interval δt. The QSSA means these predictions are constant over δt. Then we integrate to get the biomass (B) and substrate concentration (Sc) at the next time point $t + \delta t$. Therefore, the new VIN is calculated each time based on points δt in-between time. Thus we can predict the growth rate and glucose and acetate uptake (nutrients available in the environment). The four step analysis is:

1. The concentration at time t is given by the substrate concentration from the last step plus any additional substrate provided to the cell culture by an inflow, such as in a fed batch.
2. The substrate concentration is scaled for time and biomass (X) to determine the substrate availability to the cells. This can exceed the maximum uptake rate of the cells or be less than that number.
3. Use the flux balance model to evaluate the actual substrate uptake rate, which may be more or less than the substrate available as determined by step 2.
4. The concentration for the next time step is then calculated by integrating the standard differential equations:

$$\frac{dB}{dt} = gB \rightarrow B = B_o e^{gt}$$

$$\frac{dSc}{dt} = -SuB \rightarrow Sc = Sc_o \frac{X}{g}(e^{gt} - 1)$$

The additional work by Varma et al. (1994) specifies the glucose uptake rate a priori [17]. The model simulations work to predict time-dependent changes in growth, oxygen uptake, and acetate secretion. This

converse model plots uptake rates versus growth, while still achieving comparable results in vivo and in silico. The researchers used quasi steady state modeling to predict the time-dependent profiles of cell growth and metabolite concentrations in batch cultures of *E. coli* that had either a limited initial supply of glucose (left) or a slow continuous glucose supply (right diagram). A great fit is evident.

The diagrams above show the results of the model predictions (solid lines) and compare it to the experimental results (individual points). Thus, in *E. coli*, quasi steady state predictions are impressively accurate even with a model that does not account for any changes in enzyme expression levels over time. However, this model would not be adequate to describe behavior that is known to involve gene regulation. For example, if the cells had been grown on half-glucose/half-lactose medium, the model would not have been able to predict the switch in consumption from one carbon source to another. (This does occur experimentally when *E. coli* activates alternate carbon utilization pathways only in the absence of glucose.)

21.4.4 Regulation via Boolean Logic

There is a number of levels of regulation through which metabolic flux is controlled at the metabolite, transcriptional, translational, post-translational levels. FBA associated errors may be explained by incorporation of gene regulatory information into the models. One way to do this is Boolean logic. The following table describes if genes for associated enzymes are on or off in presence of certain nutrients (an example of incorporating *E. coli* preferences mentioned above):

ON no glucose(0)	ON acetate present(1)
ON glucose present(1)	OFF acetate present(1)

Therefore, one may think that the next step to take is to incorporate this fact into the models. For example, if we have glucose in the environment, the acetate processing related genes are off and therefore absent from the S matrix which now becomes dynamic as a result of incorporation of regulation into our model. In the end, our model is not quantitative. The basic regulation then describes that if one nutrient-processing enzyme is on, the other is off. Basically it is a bunch of Boolean logic, based on presence of enzymes, metabolites, genes, etc. These Boolean style assumptions are then used at every small change in time dt to evaluate the growth rate, the fluxes, and such variables. Then, given the predicted fluxes, the V_{IN} ,the V_{OUT} , and the system states, one can use logic to turn genes off and on, effectively a δS per δt. We can start putting together all of the above analyses and come up with a general approach in metabolic modeling. We can tell that if glycolysis is on, then gluconeogenesis must be off.

The first attempt to include regulation in an FBA model was published by Covert, Schilling, and Palsson in 1901 [7]. The researchers incorporated a set of known transcriptional regulatory events into their analysis of a metabolic regulatory network by approximating gene regulation as a Boolean process. A reaction was said to occur or not depending on the presence of both the enzyme and the substrate(s): if either the enzyme that catalyzes the reaction (E) is not expressed or a substrate (A) is not available, the reaction flux will be zero:

$$\text{rxn} = \text{IF (A) AND (E)}$$

Similar Boolean logic determined whether enzymes were expressed or not, depending on the currently expressed genes and the current environmental conditions. For example, transcription of the enzyme (E) occurs only if the appropriate gene (G) is available for transcription and no repressor (B) is present:

$$\text{trans} = \text{IF (G) AND NOT (B)}$$

The authors used these principles to design a Boolean network that inputs the current state of all relevant genes (on or off) and the current state of all metabolites (present or not present), and outputs a binary vector containing the new state of each of these genes and metabolites. The rules of the Boolean network were constructed based on experimentally determined cellular regulatory events. Treating reactions and enzyme/metabolite concentrations as binary variables does not allow for quantitative analysis, but this method can predict qualitative shifts in metabolic fluxes when merged with FBA. Whenever an enzyme is absent, the corresponding column is removed from the FBA reaction matrix, as was described above for knockout phenotype prediction. This leads to an iterative process:

1. Given the initial states of all genes and metabolites, calculate the new states using the Boolean network;
2. perform FBA with appropriate columns deleted from the matrix, based on the states of the enzymes, to determine the new metabolite concentrations;
3. repeat the Boolean network calculation with the new metabolite concentrations; etc. The above model is not quantitative, but rather a pure simulation of turning genes on and off at any particular time instant.

On a few metabolic reactions, there are rules about allowing organism to shift carbon sources (C1, C2).

An application of this method from the study by Covert et al.[7] was to simulate diauxic shift, a shift from metabolizing a preferred carbon source to another carbon source when the preferred source is not available. The modeled process includes two gene products, a regulatory protein RPc1, which senses (is activated by) Carbon 1, and a transport protein Tc2, which transports Carbon 2. If RPc1 is activated by Carbon 1, Tc2 will not be transcribed, since the cell preferentially uses Carbon 1 as a carbon source. If Carbon 1 is not available, the cell will switch to metabolic pathways based on Carbon 2 and will turn on expression of Tc2.

The Booleans can represent this information:

$$\text{RPc1} = \text{IF(Carbon1) Tc2} = \text{IF NOT(RPc1)}$$

Covert et al. found that this approach gave predictions about metabolism that matched results from experimentally induced diauxic shift. This diauxic shift is well modeled by the in silico analysis see above figure. In segment A, C1 is used up as a nutrient and there is growth. In segment B, there is no growth as C1 has run out and C2 processing enzymes are not yet made, since genes have not been turned on (or are in the process), thus the delay of constant amount of biomass. In segment C, enzymes for C2 turned on and the biomass increases as growth continues with a new nutrient source. Therefore, if there is no C1, C2 is used up. As C1 runs out, the organism shifts metabolic activity via genetic regulation and begins to take up C2. Regulation predicts diauxie, the use of C1 before C2. Without regulation, the system would grow on both C1 and C2 together to max biomass.

So far we have discussed using this combined FBA-Boolean network approach to model regulation at the transcriptional/translational level, and it will also work for other types of regulation. The main limitation is for slow forms of regulation, since this method assumes that regulatory steps are completed within a single time interval (because the Boolean calculation is done at each FBA time step and does not take into account previous states of the system). This is fine for any forms of regulation that act at least as fast as transcription/translation. For example, phosphorylation of enzymes (an enzyme activation process) is very fast and can be modeled by including the presence of a phosphorylase enzyme in the Boolean network.

However, regulation that occurs over longer time scales, such as sequestration of mRNA, is not taken into account by this model. This approach also has a fundamental problem in that it does not allow actual experimental measurements of gene expression levels to be inputted at relevant time points.

We do not need our simulations to artificially predict whether certain genes are on or off. Microarray expression data allows us to determine which genes are being expressed, and this information can be incorporated into our models.

21.4.5 Coupling Gene Expression with Metabolism

In practice, we do not need to artificially model gene levels, we can measure them. As discussed previousky, it is possible to measure the expressions levels of all the mRNAs in a given sample. Since mRNA expression data correlates with protein expression data, it would be extremely useful to incorporate it into the FBA. Usually, data from microarray experiments is clustered, and unknown genes are hypothesised to have function similar to the function of those known genes with which they cluster. This analysis can be faulty, however, as genes with similar actions may not always cluster together. Incorporating microarray expression data into FBA could allow an alternate method of interpretation of the data. Here arises a question, what is the relationship between gene level and flux through a reaction?

Say the reaction $A \rightarrow B$ is catalyzed by an enzyme. If a lot of A present, increased expression of the gene for the enzyme causes increased reaction rate. Otherwise, increasing gene expression level will not increase reaction rate. However, the enzyme concentration can be treated as a constraint on the maximum possible flux, given that the substrate also has a reasonable physiological limit.

The next step, then, is to relate the mRNA expression level to the enzyme concentration. This is more difficult, since cells have a number of regulatory mechanisms to control protein concentrations independently of mRNA concentrations. For example, translated proteins may require an additional activation step (e.g. phosphorylation), each mRNA molecule may be translated into a variable number of proteins before it is degraded (e.g. by antisense RNAs), the rate of translation from mRNA into protein may be slower than the time intervals considered in each step of FBA, and the protein degradation rate may also be slow. Despite these complications, the mRNA expression levels from microarray experiments are usually taken as upper bounds on the possible enzyme concentrations at each measured time point. Given the above relationship between enzyme concentration and flux, this means that the mRNA expression levels are also upper bounds on the maximum possible fluxes through the reactions catalyzed by their encoded proteins. The validity of this assumption is still being debated, but it has already performed well in FBA analyses and is consistent with recent evidence that cells do control metabolic enzyme levels primarily by adjusting mRNA levels. (In 1907, Professor Galagan discussed a study by Zaslaver et al. (1904) that found that genes required in an amino acid biosynthesis pathway are transcribed sequentially as needed [2]). This is a particularly useful assumption for including microarray expression data in FBA, since FBA makes use of maximum flux values to constrain the flux balance cone.

Colijn et al. address the question of algorithmic integration of expression data and metabolic networks [3]. They apply FBA to model the maximum flux through each reaction in a metabolic network. For example, if microarray data is available from an organism growing on glucose and from an organism growing on acetate, significant regulatory differences will likely be observed between the two datasets. V_{max} tells us what the maximum we can reach. Microarray detects the level of transcripts, and it gives an upper boundary of V_{max}.

In addition to predicting metabolic pathways under different environmental conditions, FBA and microarray experiments can be combined to predict the state of a metabolic system under varying drug treatments. For example, several TB drugs target mycolic acid biosynthesis. Mycolic acid is a major cell wall constituent. In a 1904 paper by Boshoff et al., researchers tested 75 drugs, drug combinations, and growth conditions to

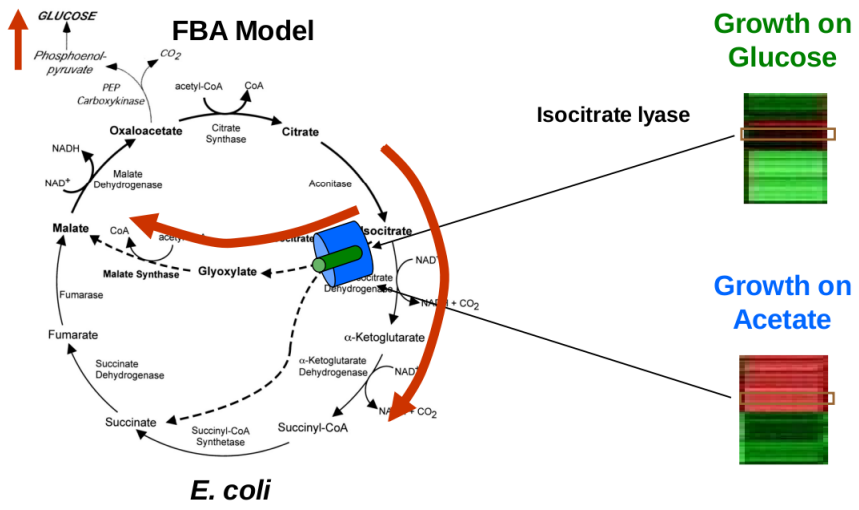


Figure 21.6: Model of Coljin et. al [3]

see what effect different treatments had on mycolic acid synthesis [9]. In 1905, Raman et al. published an FBA model of mycolic acid biosynthesis, consisting of 197 metabolites and 219 reactions [13].

The basic flow of the prediction was to take a control expression value and a treatment expression value for a particular set of genes, then feed this information into the FBA and measure the final effect on the treatment on the production of mycolic acid. To examine predicted inhibitors and enhancers, they examined significance, which examines whether the effect is due to noise, and specificity, which examines whether the effect is due to mycolic acid or overall supression/enhancement of metabolism. The results were fairly encouraging. Several known mycolic acid inhibitors were identified by the FBA. Interesting results were also found among drugs not specifically known to inhibit mycolic acid synthesis. 4 novel inhibitors and 2 novel enhancers of mycolic acid synthesis were predicted. One particular drug, Triclosan, appears to be an enhancer according to the FBA model, whereas it is currently known as an inhibitor. Further study of this particular drug would be interesting. Experimental testing and validation are currently in progress.

Clustering may also be ineffective in identifying function of various treatments. Predicted inhibitors, and predicted enhancers of mycolic acid synthesis are not clustered together. In addition, no labeled training set is required for FBA-based algorithmic classification, whereas it is necessary for supervised clustering algorithms.

21.4.6 Predicting Nutrient Source

Now, we get the idea of predicting the nutrient source that an organism may be using in an environment, by looking at expression data and looking for associated nutrient processing gene expression. This is easier, since we cant go into the environment and measure all chemical levels, but we can get expression data rather easily. That is, we try to predict a nutrient source through predictions of metabolic state from expression data, based on the assumption that organisms are likely to adjust metabolic state to available nutrients. The nutrients may then be ranked by how well they match the metabolic states.

The other way around could work too. Can I predict a nutrient given a state? Such predictions could be useful for determining the nutrient requirements of an organism with an unknown natural environment, or for determining how an organism changes its environment. (TB, for example, is able to live within the environment of a macrophage phagolysosome, presumably by altering the environmental conditions in the

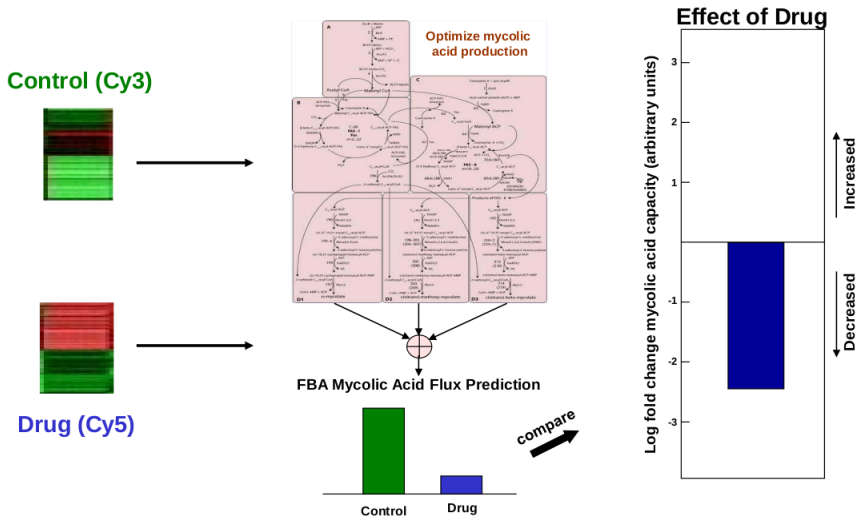


Figure 21.7: Basic flow in predicting state of a metabolic system under varing drug treatments

phagolysosome and preventing its maturation.)

We can use FBA to define a space of possible metabolic states and choose one. The basic steps are to:

- Start with max flux cone (representing best growth with all nutrients available in environment). Find optimal flux for each nutrient.
- Apply expression data set (still not knowing nutrient). This will allow you to constrain the cone shape and figure out the nutrient, which is represented as one with the closest distance to optimal solution.

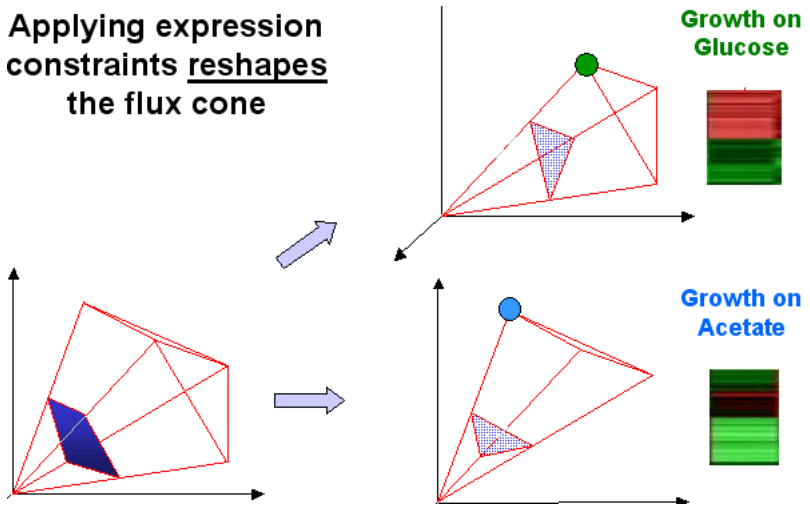


Figure 21.8: Applying expression data set allows constraining of cone shape.

In Figure 8, you may see that the first cone has a number of optimals, so the real nutrient is unknown. However, after expression data is applied, the cone is reshaped. It has only one optimal, which is still in feasible space and thereby must be that nutrient you are looking for.

As before, the measured expression levels provide constraints on the reaction fluxes, altering the shape

of the flux-balance cone (now the expression-constrained flux balance cone). FBA can be used to determine the optimal set of fluxes that maximize growth within these expression constraints, and this set of fluxes can be compared to experimentally-determined optimal growth patterns under each environmental condition of interest. The difference between the calculated state of the organism and the optimal state under each condition is a measure of how sub-optimal the current metabolic state of the organism would be if it were in fact growing under that condition.

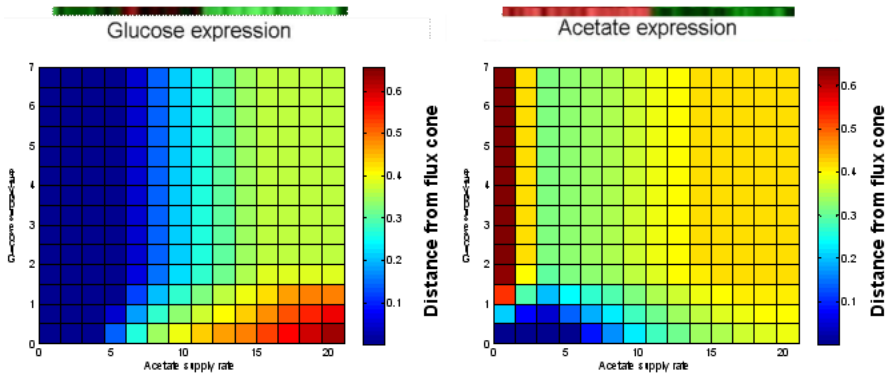


Figure 21.9: Results of nutrient source prediction experiment.

Expression data from growth and metabolism may then be applied to predict the carbon source being used. For example, consider E. coli nutrient product. We can simulate this system for glucose versus acetate. The color indicates the distance from the constrained flux cone to the optimal flux solution for that nutrient combo (same procedure described above). Then, multiple nutrients may be ranked, prioritized according to expression data. Unpublished data from Desmond Lun and Aaron Brandes provide an example of this approach.

They used FBA to predict which nutrient source E. coli cultures were growing on, based on gene expression data. They compared the known optimal fluxes (the optimal point in flux space) for each nutrient condition to the allowed optimal flux values within the expression-constrained flux-balance cone. Those nutrient conditions with optimal fluxes that remained within (or closest to) the expression-constrained cone were the most likely possibilities for the actual environment of the culture.

Results of the experiment are shown in Figure 9, where each square in the results matrices is colored based on the distance between the optimal fluxes for that nutrient condition and the calculated optimal fluxes based on the expression data. Red values indicate large distances from the expression-constrained flux cone and blue values indicate short distances from the cone. In the glucose-acetate experiments, for example, the results of the experiment on the left indicate that low acetate conditions are the most likely (and glucose was the nutrient in the culture) and the results of the experiment on the right indicate that low glucose/medium acetate conditions are the most likely (and acetate was the nutrient in the culture). When 6 possible nutrients were considered, the model always predicted the correct one, and when 18 possible nutrients were considered, the correct one was always one of the top 4 ranking predictions. These results suggest that it is possible to use expression data and FBA modeling to predict environmental conditions from information about the metabolic state of an organism.

This is important because TB uses fatty acids in macrophages in immune systems. We do not know which ones exactly are utilized. We can figure out what the TB sees in its environment as a food source and proliferation factor by analyzing what related nutrient processing genes are turned on at growth phases and such. Thereby we can figure out the nutrients it needs to grow, allowing for a potential way to kill it off by not supplying such nutrients or knocking out those particular genes.

It is easier to get expression data to see flux activity than see whats being used up in the environment by

analyzing the chemistry on such a tiny level. Also, we might not be able to grow some bacteria in lab, but we can solve the problem by getting the expression data from the bacteria growing in a natural environment and then seeing what it is using to grow. Then, we can add it to the laboratory medium to grow the bacteria successfully.

21.5 Current Research Directions

21.6 Further Reading

- Becker, S. A. and B. O. Palsson (1908). Context-Specific Metabolic Networks Are Consistent with Experiments. PLoS Computational Biology 4(5): e1000082.
  - If gene expression lower than some threshold, turn the gene off in the model.
- Shlomi, T., M. N. Cabili, et al. (1908). Network-based prediction of human tissue-specific metabolism. Nat Biotech 26(9): 1003-1010.
  - Nested optimization problem.
  - First, standard FBA
  - Second, maximize the number of enzymes whose predicted flux activity is *consistent with their measured expression level*

21.7 Tools and Techniques

- Kegg
- BioCyc
- Pathway Explorer (pathwayexplorer.genome.tugraz.at)
- Palssons group at UCSD (http://gcrg.ucsd.edu/)
- www.systems-biology.org
- Biomodels database (www.ebi.ac.uk/biomodels/)
- JWS Model Database (jjj.biochem.sun.ac.za/database/index.html)

21.8 What Have We Learned?

Bibliography

[1]

[2] Zaslaver A, Mayo AE, Rosenberg R, Bashkin P, Sberro H, Tsalyuk M, Surette MG, and Alon U. Just-in-time transcription program in metabolic pathways. *Nat. Gen*, 36:486–491, 2004.

[3] Caroline Coljin. Interpreting expression data with metabolic flux models: Predicting mycobacterium tuberculosis mycolic acid production. *PLoS Computational Biology*, 5(8), Aug 2009.

[4] Price N. D., Reed J. L., Papin J.A, Famili I., and Palsson B.O. Analysis of metabolic capabilities using singular value decomposition of extreme pathway matrices. *Biophys J.*, 84(2):794–804, Feb 2003.

[5] Gasteiger E., Gattiker A., Hoogland C. andIvanyi I., Appel R.D., , and Bairoch A. Expasy: The proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Res*, 31(13):3784–3788.

[6] J.S. Edwards, R. U. Ibarra, and B.O. Palsson. In silico predictions of e coli metabolic capabilities are consis ent with experimental data. *Nat Biotechnology*, 19:125–130, 2001.

[7] Covert M et al. Regulation of gene expression in flux balance models of metabolism. *Journal of Theoretical Biology*, 213:73–88, Nov 2001.

[8] J. Forster, I. Famili, B.O. Palsson, and J. Nielsen. Large-scale evaluation of in silico gene deletions in saccharomyces cerevisiae. *OMICS*, 7(2):193–202, 2003. PMID: 14506848.

[9] Boshoff H.I., Myers T.G., Copp B.R., McNeil M.R., Wilson M.A., and Bary C.E. The transcriptional response of mycobacterium tuberculosis to inhibitors of metabolism: novel insights into drug mechanisms of action. *J Biol Chem*, 279:40174–40184, Sep 2004.

[10] Holmberg. On the practical identifiability of microbial-growth models incorporating michaelis-menten type nonlinearities. *Mathematical Biosciences*, 62(1):23–43, 1982.

[11] Edwards J.S. and Palsson B.O. volume 97, pages 5528–5533. Proceedings of the National Academy of Sciences of the United States of America, May 2000. PMC25862.

[12] Edwards J.S., Covert M., , and Palsson B. Metabolic modeling of microbes: the flux balance approach. *Environmental Microbiology*, 4(3):133–140, 2002.

[13] Raman Karthik, Preethi Rajagopalan, and Nagasuma Chandra. Flux balance analysis of mycolic acid pathway: Targets for anti-tubercular drugs. *PLoS Computational Biology*, 1, Oct 2005.

[14] Kanehisa M., Goto S., Kawashima S., and Nakaya. From genomics to chemical genomics: new developments in kegg. *Nucleic Acids Res.*, 34, 2006.

[15] Jamshidi N. and Palsson B. Investigating the metabolic capabilities of mycobacterium tuberculosis h37rv using the in silico strain inj661 and proposing alternative drug targets. *BMC Systems Biology*, 26, 2007.

[16] Caspi R., Foerster H., Fulcher C.A., Kaipa P., Krummenacker M., Latendresse M., Paley S., Rhee S.Y., Shearer A.G., Tissier C., Walk T.C. ZhangP., and Karp P. The metacyc database of metabolic pathways and enzymes and the biocyc collection of pathway/genome databases. *Nucleic Acids Res*, 36(Suppl), 2008.

[17] A. Varma and B. O. Palsson. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type escherichia coli w3110. *Applied and Environmental Microbiology*, 60:3724–3731, Oct 1994.

CHAPTER

TWENTYTWO

THE ENCODE PROJECT: SYSTEMATIC EXPERIMENTATION AND INTEGRATIVE GENOMICS

TODO: missing *@scribe: add author*

Figures

22.1 Introduction

The human genome was sequenced in 2003, an important step in understanding the blueprint of life. However, before this information can be fully utilized, the location, identity, and function of all protein-encoding and non-protein-encoding genes must be determined. Moreover, the human genome has many other functional elements, ranging from promotors, regulatory sequences, and other factors that determine chromatin structure. These must also be determined to fully understand the human genome.

The ENCODE (Encyclopedia of DNA Elements) project aims to solve these problems by delineating all functional elements of the human genome **TODO: Reference** *@scribe: cite website*. To accomplish this goal, a consortium was formed to guide the project. The consortium aimed to advance and develop technologies for annotating the human genome with higher accuracy, completeness, and cost-effectiveness,

along with more standardization. They also aimed to develop a series of computational techniques to parse and analyze the data obtained.

To accomplish this goal, a pilot project was launched. The ENCODE pilot project aimed to study 1% of the human genome in depth, roughly from 2003 to 2007. From 2007 to 2012, the ENCODE project ramped up to annotate the entire genome. Finally, from 2012 onwards, the ENCODE project aims further increases in all dimensions: deeper sequencing, more assays, more transcription factors, etc.

This chapter will describe some of the experimental and computational techniques used in the ENCODE project.

22.2 Experimental Techniques

The ENCODE project used a wide range of experimental techniques, ranging from RNA-seq, CAGE-seq, Exon Arrays, MAINE-seq, Chromatin ChIP-seq, DNase-seq, and many more.

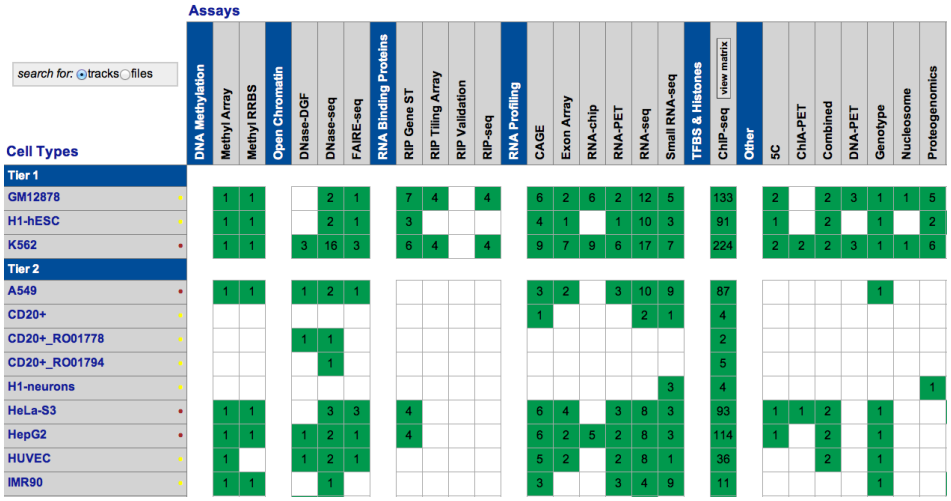


Figure 22.1: Snapshot of ENCODE project experiment matrix.

One of the most important techniques used was ChIP-seq (chromatin immunoprecipitation followed by sequencing). The first step in a ChIP experiment is to target DNA fragments associated with a specific protein. This is done by using an anti-body that targets the specific protein and is used to immunoprecipitate the DNA-protein complex. The final step is to assay the DNA. This will determine the sequences bound to the proteins.

ChIP-seq has several advantages over previous techniques (e.g. ChIP-chip). For example, ChIP-seq has single nucleotide resolution and its alignability increases with read length. However, ChIP-seq has several disadvantages. Sequencing errors tend to increase substantially near the end of reads. Also, with low number of reads, sensitivity and specificity tend to decrease when detecting enriched regions. Both of these problems arise when processing the data and many of the computational techniques seek to rectify this.

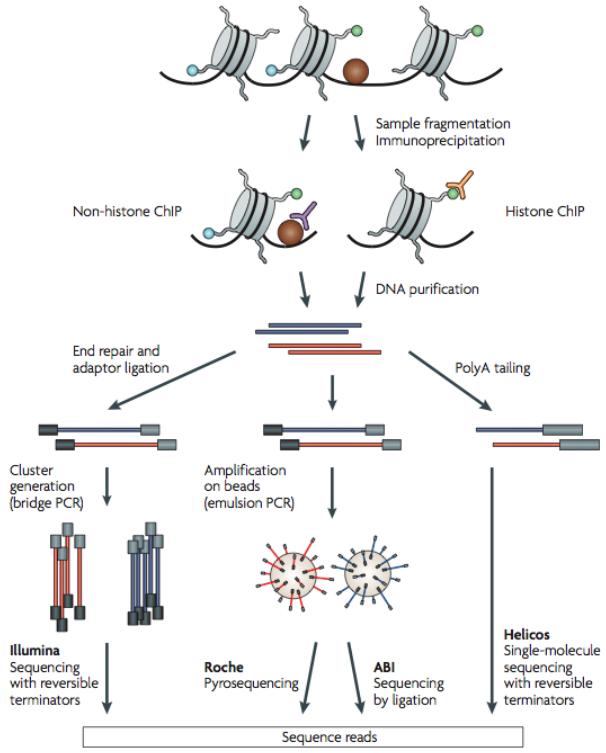


Figure 22.2: Overview of Chip-seq [3]

22.3 Computational Techniques

This section will focus on techniques on processing raw data from the ENCODE project. Before ENCODE data can be analyzed (e.g. for motif discovery, co-association analysis, signal aggregation over elements, etc), the raw data must be processed.

Even before the data is processed, some quality control is applied. Quality control is needed for several reasons. Even without anti-bodies, reads are not uniformly-scattered. The biological reasons include non-uniform fragmentation of the genome, open chromatin regions fragmenting easier, and repetitive sequences over-collapsed in assembled genomes. The ENCODE project corrected for these biases in several ways. Portions of the DNA were removed before the ChIP step, removing large portions of unwanted data. Control experiments were also conducted without the use of anti-bodies. Finally, fragment input DNA sequence reads were used as a background.

Because of inherent noise in the ChIP-seq process, some reads will be of lower quality. Using a read quality metric, reads below a threshold were thrown out.

Shorter reads (and to a lesser extent, longer reads) can map to exactly one location (uniquely mapping), multiple locations (repetitive mapping), or no locations at all (unmappable) in the genome. There are many potential ways to deal with repetitive mapping, ranging from probabilistically spreading the read to use an EM approach. However, since the ENCODE project aims to be as correct as possible, it does not assign repetitive reads to any location.

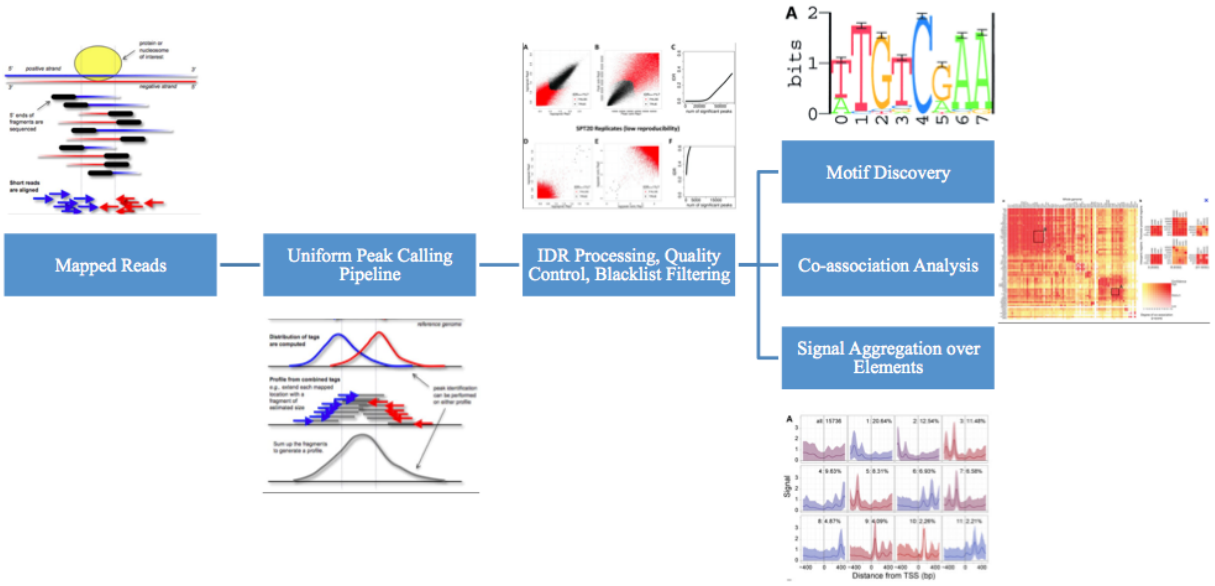


Figure 22.3: ENCODE uniform Processing Pipeline

If a sample does not contain sufficient DNA and/or if it is over-sequenced, you will simply be repeatedly sequencing PCR duplicates of a restricted pool of distinct DNA fragments. This is known a low-complexity library and is not desirable. To solve this problem, a histogram with the number of duplicates is created and samples with a low non-redundant fraction (NRF) are thrown out.

ChIP-seq randomly sequences from one end of each fragment, so to determine which reads came from which segment, typically strand cross-correlation analysis is used [Fig. 04]. To accomplish this, the forward and and reverse strand signals are calculated. Then, they are sequentially shifted towards each other. At every step, the correlation is calculated. At the fragment length offset f, the correlation peaks. f is the length at which ChIP DNA is fragmented. Using further analysis, we can determine that we should have a high absolute cross-correlation at fragment length, and high fragment length cross-correlation relative to read-length cross-correlation. The RSC (Relative Strand Correlation) should be greater than 1.

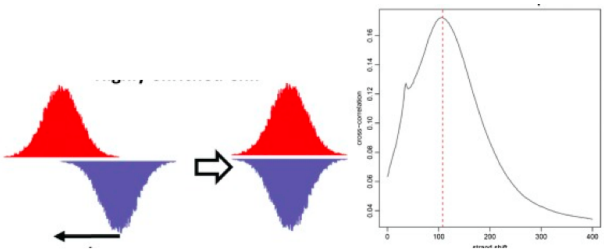


Figure 22.4: Shifting forward and reverse strand signals, Cross Correlation plot

$$RSC = \frac{CC_{fragment} - min(CC)}{CC_{readlength} - min(CC)} \tag{22.1}$$

Once quality control is applied, the data is further processed to determine actual areas of enrichment. To accomplish this, the ENCODE project used a modified version of peak calling. There are many existing peak

calling algorithms, but the ENCODE project used MACS and PeakSeq, as they are deterministic. However, it is not possible to set a uniform p-value or false discovery rate (FDR) constant. The FDR and p-value depends on ChIP and input sequencing depth, the binding ubiquity of the factor, and is highly unstable. Moreover, different tools require different values.

The ENCODE project uses replicates (of the same experiment) and combines the data to find more meaningful results. Simple solutions have major issues: taking the union of the peaks keeps garbage from both, the intersection is too stringent and throws away good peaks, and taking the sum of the data does not exploit the independence of the datasets. Instead, the ENCODE project uses the independent discovery rate (IDR). The key idea is that true peaks will be highly ranked in both replicates. Thus, to find significant peaks, the peaks are considered in rank order, until ranks are no longer correlated.

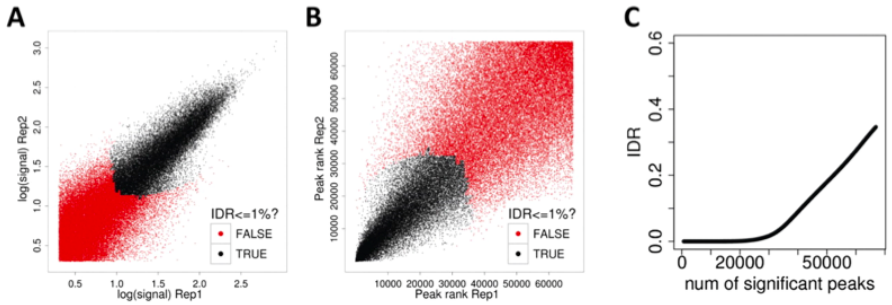


Figure 22.5: IDR to assess reproducibility of CHIP-seq datasets. Scatter plots display signal scores of peaks that overlap in each replicate pair. (A,B) results for high quality replicate. (C) Estimated IDR for varying rank thresholds. [1]

The cutoff could be different for the two replicates and actual peaks included may differ between replicates. It is modeled as a Gaussian mixture model, which can be fit via an EM-like algorithm. Using IDR leads to higher consistence between peak callers. This is because FDR only relies on enrichment over input, IDR exploits replicates. Also, using sampling methods, if there is only one replicate, the IDR pipeline can still be used with pseudo-replicates.

TODO: figure *@scribe: Expand on the following topics; Aggregation Analysis, Predictive models of TF co-association and gene expression, Scaling and Saturaion Analysis*

22.4 Current Research Directions

The ENCODE project is still ongoing. Using saturation techniques, we believe we only have discovered a maximum 50% of elements. This number is likely to be lower due to inaccessible cell types and other factors. Also, several cell types are extremely rare and difficult to access, so sequencing data from these cell types is another challenge.

In computational frontiers, the ENCODE project has produced an enormous amount of raw data. Similar to how the full sequence of the human genome unleashed a series of computational projects, the ENCODE data can be used for a variety of computational projects.

22.5 Further Reading

The Nature site with ENCODE papers is available at http://www.nature.com/encode/.

The official ENCODE portal is http://encodeproject.org/ENCODE/.

To browse ENCODE data, visit http://encodeproject.org/cgi-bin/hgHubConnect.

Data processing tools for ENCODE data are available at http://encodeproject.org/ENCODE/analysis.html.

22.6 Tools and Techniques

ENCODE data mining, http://genome.ucsc.edu/cgi-bin/hgTables?db=hg18&hgta_group=regulation&hgta_track=wgEncodeHudsonalphaChipSeq

ENCODE data visualization, http://genome.ucsc.edu/cgi-bin/hgTracks?hgS_doOtherUser=submit&hgS_otherUserName=Kate&hgS_otherUserSessionName=encodePortalSession

Software and resources for rnalyzing ENCODE data, http://genome.ucsc.edu/ENCODE/analysisTools.html

Software tools used to create the ENCODE resource, http://genome.ucsc.edu/ENCODE/encodeTools.html

22.7 What Have We Learned?

This chapter provides an overview the ENCODE project which aims to annotate the entire human genome. It collects DNA sequences using various experimental techniques such as CHIP-seq, RNA-seq, and CAGE-seq. After the data has been obtained it needs to be processed before attempting analysis. The data goes through a number of steps; quality control, peak calling, IDR processing, and blacklist filtering. Once the accuracy of the data has been ensured other analysis can be done in the form of motif discovery, co-association analysis, and signal aggregation over elements.

TODO: figure *@scribe: there are extra figures in 'images' folder.*

Bibliography

[1] S G Landt, G K Marinov, A Kundaje, P Kheradpour, F Pauli, S Batzoglou, B E Bernstein, P Bickel, J B Brown, P Cayting, Y Chen, G DeSalvo, C Epstein, K I Fisher-Aylor, G Euskirchen, M Gerstein, J Gertz, A J Hartemink, M M Hoffman, V R Iyer, Y L Jung, S Karmakar, M Kellis, P V Kharchenko, Q Li, T Liu, X S Liu, L Ma, A Milosavljevic, R M Myers, P J Park, M J Pazin, M D Perry, D Raha, T E Reddy, J Rozowsky, N Shoresh, A Sidow, M Slattery, J A Stamatoyannopoulos, M Y Tolstorukov,

K P White, S Xi, P J Farnham, J D Lieb, B J Wold, and M Snyder. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome Research*, 22(9):1813–1831, 2012.

[2] Philippe Lefrançois, Ghia M Euskirchen, Raymond K Auerbach, Joel Rozowsky, Theodore Gibson, Christopher M Yellman, Mark Gerstein, and Michael Snyder. Efficient yeast ChIP-Seq using multiplex short-read DNA sequencing. *BMC Genomics*, 10(1):37, 2009.

[3] P J Park. ChIP-seq: advantages and challenges of a maturing technology. *Nature reviews. Genetics*, 10(10):669–680, October 2009.

CHAPTER

TWENTYTHREE

PHARMACOGENOMICS

TODO: missing *@scribe: add author*

23.1 Introduction

23.2 Current Research Directions

23.3 Further Reading

23.4 Tools and Techniques

23.5 What Have We Learned?

Bibliography

CHAPTER

TWENTYFOUR

SYNTHETIC BIOLOGY

TODO: missing *@scribe: add author*

Figures

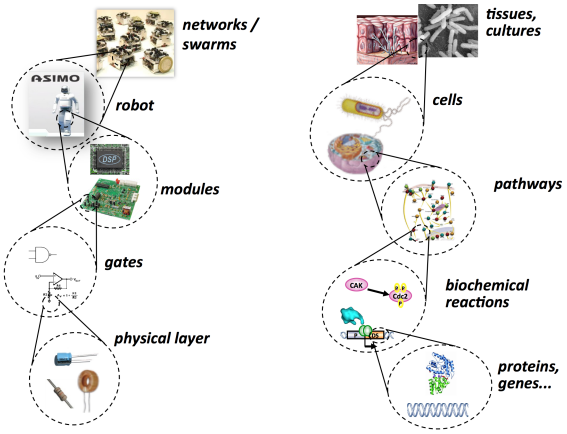
24.1 Introduction

A cell is like robot in that it needs to be able to sense it surroundings and internal state, perform computations and make judgments, and complete a task or function. The emerging discipline of synthetic biology aims to make control of biological entities such as cells and proteins similar to designing a robot. Synthetic biology combines technology, science, and engineering to construct biological devices and systems for useful purposes including solutions to world problems in health, energy, environment and, security.

Synthetic biology involves every level of biology, from DNA to tissues. Synthetic biologist aims to create layers of biological abstraction like those in digital computers in order to create biological circuits and programs efficiently. One of the major goals in synthetic biology is development of a standard and well-defined set of tools for building biological systems that allows the level of abstraction available to electrical engineers building complex circuits to be available to synthetic biologists.

Synthetic biology is a relatively new field. The size and complexity of synthetic genetic circuits has so far been small, on the order of six to eleven promoters. Synthetic genetic circuits remain small in total size

Figure 24.1: The layers of abstraction in robotics compared with those in biology (credit to Ron Weiss).



(10^3 - 10^5 base pairs) compared to size of the typical genome in a mammal or other animal (10^5 - 10^7 base pairs) as well.

One of the first milestones in synthetic biology occurred in 2000 with the repressilator. The repressilator [2] is a synthetic genetic regulatory network which acts like an electrical oscillator system with fixed time periods. A green fluorescent protein was expressed within *E. coli* and the fluorescence was measured over time. Three genes in a feedback loop were set up so that each gene repressed the next gene in the loop and was repressed by the previous gene.

Figure 24.2: The repressilator genetic regulator network.

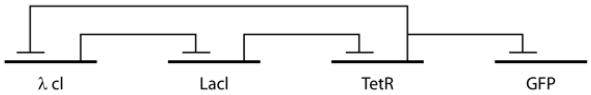
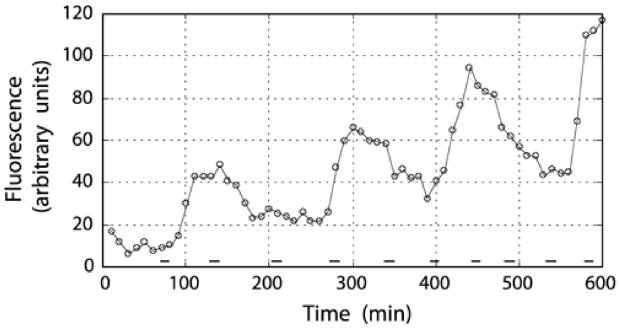


Figure 24.3: Fluorescence of a single cell with the repressilator circuit over a period of 10 hours.

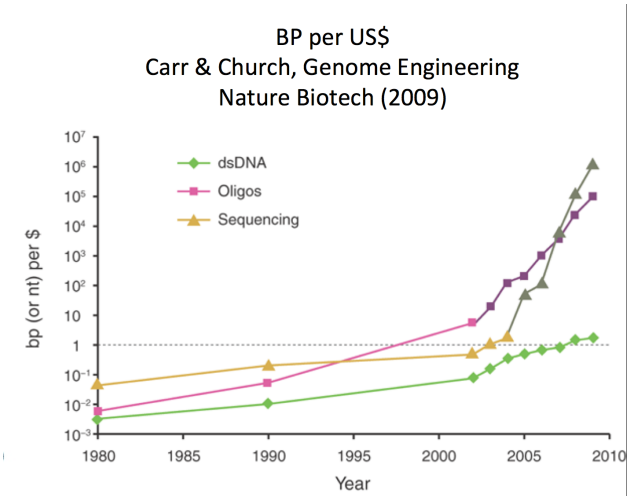


The repressilator managed to produce periodic fluctuations in fluorescence. It served as one of the first triumphs in synthetic biology. Other achievements in the past decade include programmed bacterial population control, programmed pattern formation, artificial cell-cell communication in yeast, logic gate creation by chemical complementation with transcription factors, and the complete synthesis, cloning, and assembly of a bacterial genome.

24.2 Current Research Directions

Encoding functionality in DNA is one way synthetic biologists program cells. As the price of sequencing and synthesis of DNA continues to decrease, coding DNA strands has become more feasible. In fact, the number of base pairs that can be synthesized per US$ has increased exponentially, akin to Moore's Law.

Figure 24.4: Cost of synthesizing a base pair versus US dollar



This has made the process of designing, building, and testing biological circuits much faster and cheaper. One of the major research areas in synthetic biology is the creation of fast, automated synthesis of DNA molecules and the creation of cells with the desired DNA sequence. The goal of creating a such a system is speeding up the design and debugging of making a biological system so that synthetic biological systems can be prototyped and tested in a quick, iterative process.

Synthetic biology also aims to develop abstract biological components that have standard and well-defined behavior like a part an electrical engineer might order from a catalogue. To accomplish this, the Registry of Standard Biological Parts (http://partsregistry.org) [4] was created in 2003 and currently contains over 7000 available parts for users. The research portion of creating such a registry includes the classification and description of biological parts. The goal is to find parts that have desirable characteristics such as:

Orthogonality Regulators should not interfere with each other. They should be independent.

Composability Regulators can be fused to give composite function.

Connectivity Regulators can be chained together to allow cascades and feedback.

Homogeneity Regulators should obey very similar physics. This allows for predictability and efficiency.

Synthetic biology is still developing, and research can still be done by people with little background in the field. The International Genetically Modified Machine (iGEM) Foundation (http://igem.org) [3] created the iGEM competition where undergraduate and high school students compete to design and build biological systems that operate within living cells. The student teams are given a kit of biological parts at the beginning of the summer and work at their own institutions to create biological system. Some interesting projects include:

Arsenic Biodetector The aim was to develop a bacterial biosensor that responds to a range of arsenic concentrations and produces a change in pH that can be calibrated in relation to arsenic concentration. The team's goal was to help many under-developed countries, in particular Bangladesh, to detect arsenic contamination in water. The proposed device was intended be more economical, portable and easier to use in comparison with other detectors.

BactoBlood The UC Berkeley team worked to develop a cost-effective red blood cell substitute constructed from engineered E. coli bacteria. The system is designed to safely transport oxygen in the bloodstream without inducing sepsis, and to be stored for prolonged periods in a freeze-dried state.

E. Chromi The Cambridge team project strived to facilitate biosensor design and construction. They designed and characterised two types of parts - Sensitivity Tuners and Colour Generators – E. coli engineered to produce different pigments in response to different concentrations of an inducer. The availability of these parts revolutionized the path of future biosensor design.

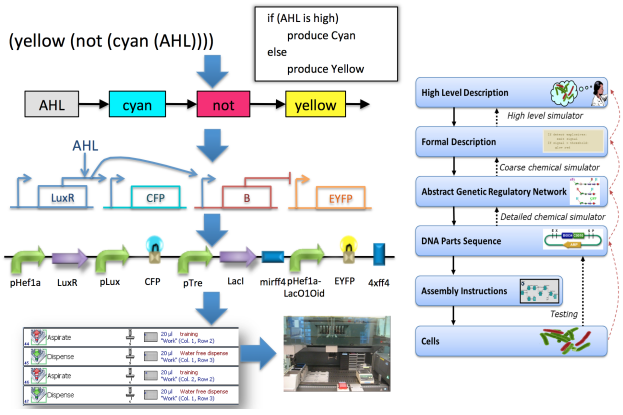
24.3 Further Reading

24.4 Tools and Techniques

Synthetic biology combines many fields, and the techniques used are not particular to synthetic biology. Much like the process of solving other engineering problems, the process of creating a useful biological system has designing, building, testing, and improving phases. Once a design or statement of the desired properties of a biological system are created, the problem becomes finding the proper biological components to build such a system.

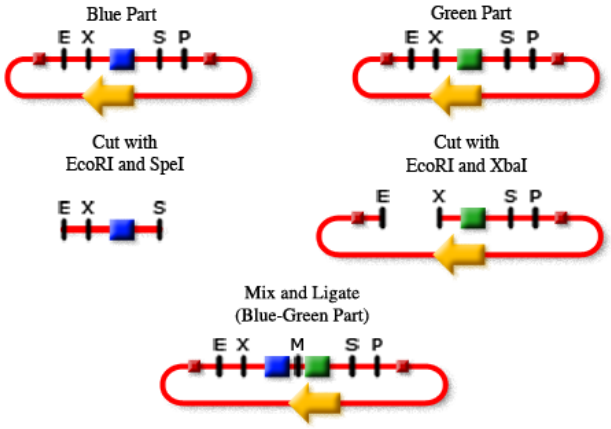
BioCompiler [1] is a tool developed to allow the programming of biological circuits using a high-level programming language. One can write programs in a language similar to LISP and compile their program into a biological circuit. BioCompiler uses a process similar to that of a compiler for a programming language. It uses a human-written program as a high-level description of the genetic circuit, then generates a formal description of the program. From there, it looks up abstract genetic regulatory network pieces that can be combined to create the genetic circuit and goes through its library of DNA parts to find appropriate sequences to match the functionality of the abstract genetic regulatory network pieces. Assembly instructions can then be generated for creating cells with the appropriate genetic regulatory network.

Figure 24.5: An example of a BioCompiler program and the process of actualizing it (credit to Ron Weiss)



BioBrick standard biologic parts (biobricks.org)are another tool used in synthetic biology. Similar to the parts in the Registry of Standard Biological Parts, BioBrick standard biological parts are DNA sequences of defined structure and function. Each BioBrick part is a DNA sequence held together in a circular plasmid. At either end of the BioBrick contains a known and well-defined sequence with restriction enzymes that can cut open the plasmid at known positions. This allows for the creation of larger BioBrick parts by chaining together smaller ones. Some competitors in the iGEM competition used BioBrick systems to develop an *E. coli* line that produced scents such as banana or mint.

Figure 24.6: An example of combining BioBrick Pieces taken from http://2006.igem.org/wiki/index.php/Standard_Assembly



24.5 What Have We Learned?

Synthetic biology is an emerging disciplines that aims to create useful biological systems to solve problems in energy, medicine, environment, and many more fields. Synthetic biologists attempt to use abstraction to enable them to build more complex systems from simpler ones in a similar way to how a software engineer or an electrical engineer would make a computer program or a complex circuit. The Registry of Standard Biological Parts and BioBrick standard biological parts aim to characterize and standardize biological pieces just as one would a transistor or logic gate to enable abstraction. Tools such as BioCompiler allow people to describe a genetic circuit using a high-level language and actually build a genetic circuit with the described functionality. Synthetic biology is still new, and research can be done by those unfamiliar with the field, as demonstrated by the iGEM competition.

Bibliography

[1] J. Beal and J. Bachrach. Cells are plausible targets for high-level spatial languages, 2008.

[2] M. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.

[3] iGEM. igem: Synthetic biology based on standard parts, December 2012.

[4] Registry of Standard Biological Parts. Registry of standard biological parts, December 2012.

Part IV

Phylogenomics and Population Genomics

CHAPTER

TWENTYFIVE

MOLECULAR EVOLUTION AND PHYLOGENETICS

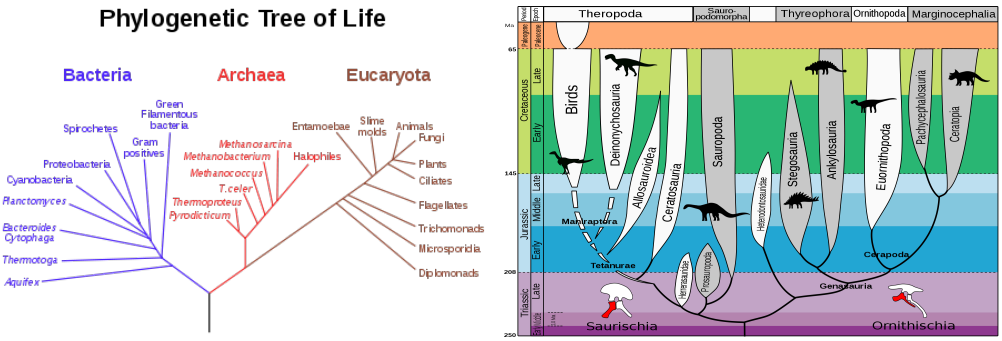
Scribed by Andrew Cooper, Stephanie Chang, and Stephen Serene (2012)
Akashnil Dutta (2011)
Albert Wang and Mashaal Sohail (2010)
Guo-Liang Chew and Sara Baldwin (2009)

Figures

25.1 Introduction

Phylogenetics is the study of relationships among a set of objects having a common origin, based on the knowledge of the individual traits of the objects. Such objects may be species, genes, or languages, and their corresponding traits may be morphological characteristics, sequences, words etc. In all these examples the objects under study change gradually with time and diverge from common origins to present day objects.

In Biology, phylogenetics is particularly relevant because all biological species happen to be descendants of a single common ancestor which existed approximately 3.5 to 3.8 billion years ago. Throughout the passage of time, genetic variation, isolation and selection have created the great variety of species that we observe today. Not just speciation however, but extinction has also played a key role in shaping the biosphere as we see today. Studying the ancestry between different species is fundamentally important to biology because they shed much light in understanding different biological functions, genetic mechanisms as well as the process of evolution itself.



(a) Tree of Life (b) Divergence and Extinction of Dinosaurs

Figure 25.1: Evolutionary History of Life

25.2 Basics of Phylogeny

25.2.1 Trees

A tree is a mathematical representation of relationships between objects. A general tree is built from nodes and edges. Each node represents an object, and each edge represents a relationship between two nodes. In the case of phylogenetic trees, we represent evolution using trees. In this case, each node represents a divergence event between two ancestral lineages, the leaves denote the set of present objects and the root represents the common ancestor.

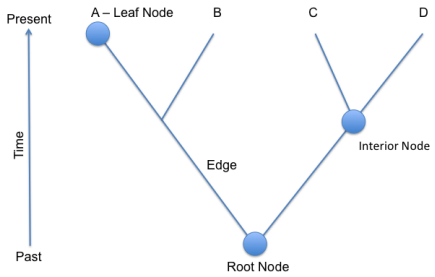


Figure 25.2: Defining tree terminology. A tree of branching nodes is depicted with leaves at the top and the root on the bottom. Time continues upward, toward the leaves.

However, sometimes more information is reflected in the branch lengths, such as time elapsed or the amount of dissimilarity. According to these differences, biological phylogenetic trees may be classified into three categories:

Cladogram: gives no meaning to branch lengths; only the sequence and topology of the branching matters.

Phylogram: Branch lengths are directly related to the amount of **genetic change**. The longer the branch of a tree, the greater the amount of phylogenetic change that has taken place. The leaves in this tree may not necessarily end on the same vertical line, due to different rates of mutation.

Chronogram (ultrametric tree): Branch lengths are directly related to **time**. The longer the branches of a tree, the greater the amount of time that has passed. The leaves in this tree necessarily end on the same vertical line (i.e. they are the same distance from the root), since they are all in the present unless extinct species were included in the tree. Although there is a correlation between branch lengths and genetic distance on a chronogram, they are not necessarily exactly proportional because evolution rates / mutation rates are not constant. Some species evolve and mutate faster than others, and some historical time periods foster faster rates of evolution than others.

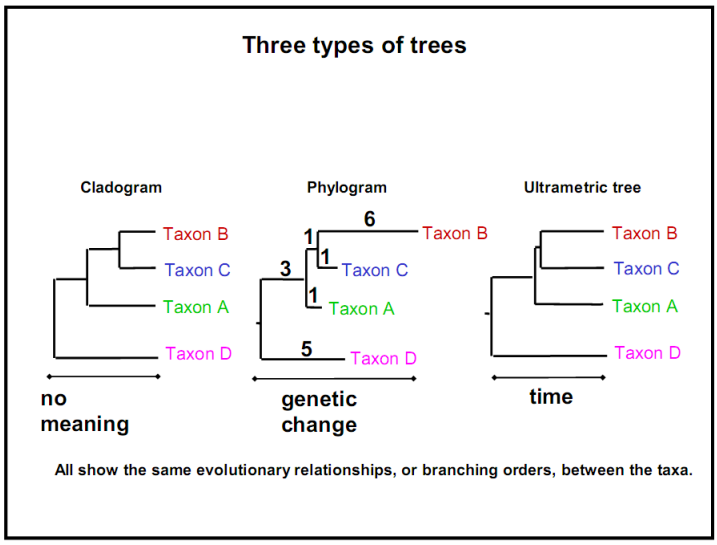


Figure 25.3: Three types of trees.

25.2.2 Traits

A trait is any characteristic that an object or species possesses. In humans, an example of a trait may be bipedalism (the ability to walk upright) or the opposable thumb. Another human trait may be a specific DNA sequence that humans possess. The first examples of physical traits are called **morphological traits**, while the latter DNA traits are called **sequence traits**. Each has its advantages and disadvantages to study. All methods for tree-reconstruction rely on studying the occurrence of different traits in the given objects. In traditional phylogenetics the morphological data of different species were used for this purpose. In modern methods, genetic sequence data is used instead. Each has its advantages and disadvantages.

Morphological Traits: Arise from empirical evaluation of physical traits. This can be advantageous because physical characteristics are very easy to quantify and understand for everyone, scientists and

children alike. The disadvantages to this approach are that we can only evaluate a small set of traits, such as hair, nails, hoofs, teeth, etc. Further, these traits only allow us to build species. Finally, it is much easier to be "tricked" by convergent evolution. Species that diverged millions of years ago may converge again on the few traits that are observable to scientists, giving a false representation of how closely related the species are.

Sequence Traits: Are discovered by studying the genomes of different species. This approach can be advantageous because it creates much more data and allows scientists to create gene trees in addition to species trees. The primary difficulty with this approach is that DNA is only built from 4 bases, so back mutations are frequent. In this approach, scientists must reconcile the signals of a large number of ill-behaved traits as opposed to that of a small number of well-behaved traits in the traditional approach. The rest of the chapter will focus principally on tree building from gene sequences.

Since this approach deals with comparing between pairs of genes, it is useful to understand the concept of **homology**: A pair of genes are called **paralogues** if they diverged from a duplication event, and **orthologues** if they diverged from a speciation event.

FAQ

Q: Would it be possible to use extinct species' DNA sequences?

A: Current technologies only allow for usage of extant sequences. However, there have been a few successes in using extinct species' DNA. DNA from frozen mammoths have been collected and are being sequences but due to DNA breaking down over time and contamination from the environment, it is very hard to extract correct sequences.

25.2.3 Methods for Tree Reconstruction

Once we have found genetic data for a set of species, we are interested in learning how those species relate to one another. Since we can, for the most part, only obtain DNA from living creatures, we must infer the existence of ancestors of each species, and ultimately infer the existence of a common ancestor. This is a challenging problem, because very limited data is available. The following sections will explore the modern methods for inferring ancestry from sequence data. They can be classified into two approaches, distance based methods and character based methods.

Distance based approaches take two steps to solve the problem, i.e. to quantify the amount of mutation that separates each pair of sequences (which may or may not be proportional to the time since they have been separated) and to fit the most likely tree according to the pair-wise distance matrix. The second step is usually a direct algorithm, based on some assumtions, but may be more complex.

Charecter based approaches instead try to find the tree that best explains the observed sequences. As opposed to direct reconstruction, these methods rely on tree proposal and scoring techniques to perform a heuristic search over the space of trees.

Did You Know?

Occam's Razor, as discussed in previous chapters, does not always provide the most accurate hypothesis. In many cases during tree reconstruction, the simplest explanation is not the most probable. For example, a set of possible ancestries may be possible, given some observed data. In this case, the simplest ancestry may not be correct if a trait arose independently in two seperate lineages. This issue will be considered in a later section.

25.3 Distance Based Methods

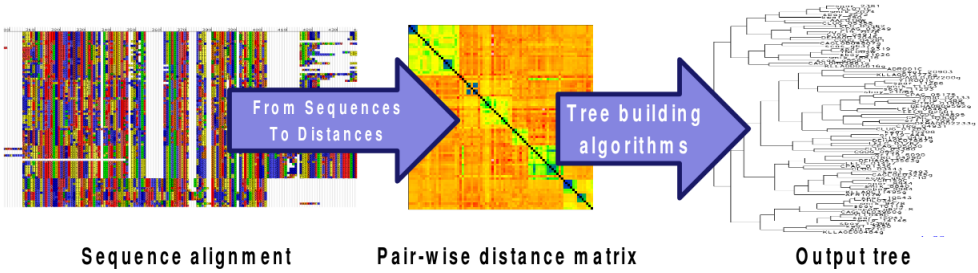


Figure 25.4: The two steps of distance based phylogenetic reconstruction.

The distance based models sequester the sequence data into pairwise distances. This step loses some information, but sets up the platform for direct tree reconstruction. The two steps of this method are hereby discussed in detail.

25.3.1 From alignment to distances

In order to understand how a distance-based model works, it is important to think about what distance means when comparing two sequences. There are three main interpretations.

Nucleotide Divergence is the idea of measuring distance between two sequences based on the number of places where nucleotides are not consistent. This assumes that evolution happens at a uniform rate across the genome, and that a given nucleotide is just as likely to evolve into any of the other three nucleotides. Although it has shortcomings, this is often a great way to think about it.

Transitions and Transversions This is similar to nucleotide divergence, but it recognizes that A-G and T-C substitutions are most frequent. Therefore, it keeps two parameters, the probability of a transition and the probability of a transversion.

Synonymous and non-synonymous substitutions This method keeps tracks of substitutions that affect the coded amino-acid by assuming that substitutions that do not change the coded protein will not be selected against, and will thus have a higher probability of occurring than those substitutions which do change the coded amino acid.

The naive way to interpret the separation between two sequences may be simply the number of mismatches, as described by nucleotide divergence above. While this does provide us a distance metric (i.e. $d(a,b) + d(b,c) \geq d(a,c)$) this does not quite satisfy our requirements, beecause we want **additive distances**, i.e. those that satisfy $d(a,b) + d(b,c) = d(a,c)$ for a path $a \to b \to c$ of evolving sequence, because the amount of mutations accumulated along a path in the tree should be the sum of that of its individual components. However, the naive mismatch fraction do not always have this property, because this quantity is bounded by 1, while the sum of individual components can easily exceed 1.

The key to resolving this paradox is **back-mutations**. When a large number of mutations accumulate on a sequence, not all the mutations introduce new mismatches, some of them may occur on already mutated base pair, resulting in the mismatch score remaining the same or even decreasing. For small mismatch-scores however, this effect is statistically insignificant, because there are vastly more identical pairs than

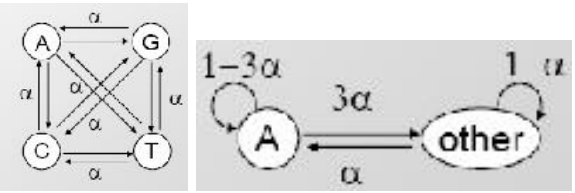
mismatching pairs. However, for sequences separated by longer evolutionary distance, we must correct for this effect. The Jukes-Cantor model is one such simple markov model that takes this into account.

Jukes-Cantor distances

To illustrate this concept, consider a nucleotide in state 'A' at time zero. At each time step, it has a probability 0.7 of retaining its previous state and probability 0.1 of transitioning to each of the other three states. The probability $P(B|t)$ of observing state (base) B at time t essentially follows the recursion

$$P(B|t+1) = 0.7P(B|t) + 0.1\sum_{b \neq B} P(b|t) = 0.1 + 0.6P(B|t)$$

.



(a) 4 States (b) 2 States

Figure 25.5: Markov chain accounting for back mutations

If we plot $P(B|t)$ versus t, we observe that the distribution starts off as concentrated at the state 'A' and gradually spreads over to the rest of the states, eventually going towards an equilibrium of equal probabilities. This progression makes sense, intuitively. Over millions of years, species can evolve so dramatically that they no longer resemble their ancestors. At that extreme, a given base location in the ancestor is just as likely to have evolved to any of the four possible bases in that location over time.

time:-	0	1	2	3	4
A	1	0.7	0.52	0.412	0.3472
C	0	0.1	0.16	0.196	0.2196
G	0	0.1	0.16	0.196	0.2196
T	0	0.1	0.16	0.196	0.2196

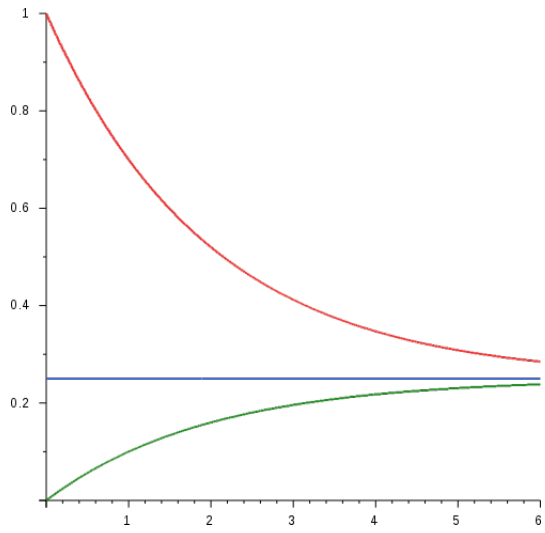


Figure 25.6: The y axis denotes probability of observing the bases - A(red), others(green). x axis denotes time.

The essence of the Jukes Cantor model is to backtrack t, the amount of time elapsed from the fraction of altered bases. Conceptually, this is just inverting the x and y axis of the green curve. To model this quantitatively, we consider the following matrix $S(t)$ which denotes the respective probabilities $P(x|y, \Delta t)$ of observing base x given a starting state of base y in time Δt.

$$S(\Delta t) = \begin{pmatrix} P(A|A,\Delta t) & P(A|G,\Delta t) & \cdots & P(A|T,\Delta t) \\ P(G|A,\Delta t) & \cdots & & \cdots \\ \cdots & & & \cdots \\ P(T|A,\Delta t) & \cdots & \cdots & P(T|T\Delta T) \end{pmatrix}$$

We can assume this is a stationary markov model, implying this matrix is multiplicative, i.e.

$$S(t_1 + t_2) = S(t_1)S(t_2)$$

For a very short time ϵ, we can assume that there is no second order effect, i.e. there isn't enough time for two mutations to occur at the same nucleotide. So the probabilities of cross transitions are all proportional to ϵ. Further, in Jukes Cantor model, we assume that all the transition rates are same from each nucleotide to another nucleotide. Hence, for a short time ϵ

$$S(\epsilon) = \begin{pmatrix} 1-3\alpha\epsilon & \alpha\epsilon & \alpha\epsilon & \alpha\epsilon \\ \alpha\epsilon & 1-3\alpha\epsilon & \alpha\epsilon & \alpha\epsilon \\ \alpha\epsilon & \alpha\epsilon & 1-3\alpha\epsilon & \alpha\epsilon \\ \alpha\epsilon & \alpha\epsilon & \alpha\epsilon & 1-3\alpha\epsilon \end{pmatrix}$$

At time t, the matrix is given by

$$S(t) = \begin{pmatrix} r(t) & s(t) & s(t) & s(t) \\ s(t) & r(t) & s(t) & s(t) \\ s(t) & s(t) & r(t) & s(t) \\ s(t) & s(t) & s(t) & r(t) \end{pmatrix}$$

From the equation $S(t+\epsilon) = S(t)S(\epsilon)$ we obtain

$$r(t+\epsilon) = r(t)(1-3\alpha\epsilon) + 3\alpha\epsilon s(t) \text{ and } s(t+\epsilon) = s(t)(1-\alpha\epsilon) + \alpha\epsilon r(t))$$

Which rearrange as the coupled system of differential equations

$$r'(t) = 3\alpha(-r(t) + s(t)) \text{ and } s'(t) = \alpha(r(t) - s(t))$$

With the initial conditions $r(0) = 1$ and $s(0) = 0$. The solutions can be obtained as

$$r(t) = \frac{1}{4}(1 + 3e^{-4\alpha t}) \text{ and } s(t) = \frac{1}{4}(1 - e^{-4\alpha t})$$

Now, in a given alignment, if we have the fraction f of the sites where the bases differ, we have:

$$f = 3s(t) = \frac{3}{4}(1 - e^{-4\alpha t})$$

implying

$$t \propto -\log\left(1 - \frac{4f}{3}\right)$$

To agree asymptotically with f, we set the evolutionary distance d to be

$$\boxed{d = -\frac{3}{4}\log\left(1 - \frac{4f}{3}\right)}$$

Note that distance is approximately proportional to f for small values of f and asymptotically approaches infinity when $f \to 0.75$. Intuitively this happens because after a very long period of time, we would expect the sequence to be completely random and that would imply about three-fourth of the bases mismatching with original. But the uncertainty values of the Jukes-Cantor distance also becomes very large when f approaches 0.75.

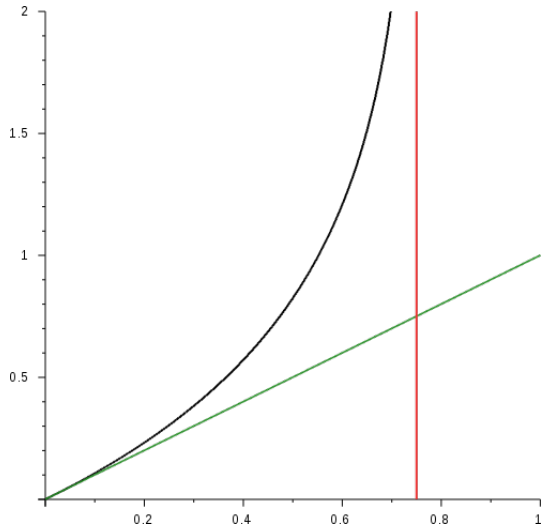


Figure 25.7: Fraction of altered bases (x axis) versus the Jukes Cantor distance(y axis).
Black line denotes the curve, green is the trend line for small values of f while the red line denotes the asymptotic boundary.

Other Models

The Jukes Cantor model is the simplest model that gives us theoretically consistent additive distance model. However, it is a one-parameter model that assumes that the mutations from each base to a different base has the same chance. But, changes between AG or between TC are more likely than changes across them. The first type of substitution is called transitions while the second type is called transversions. The Kimura model has two parameters which take this into account. There are also many other modifications of this distance model that takes into account the different rates of transitions and transversions etc. that are depicted below.

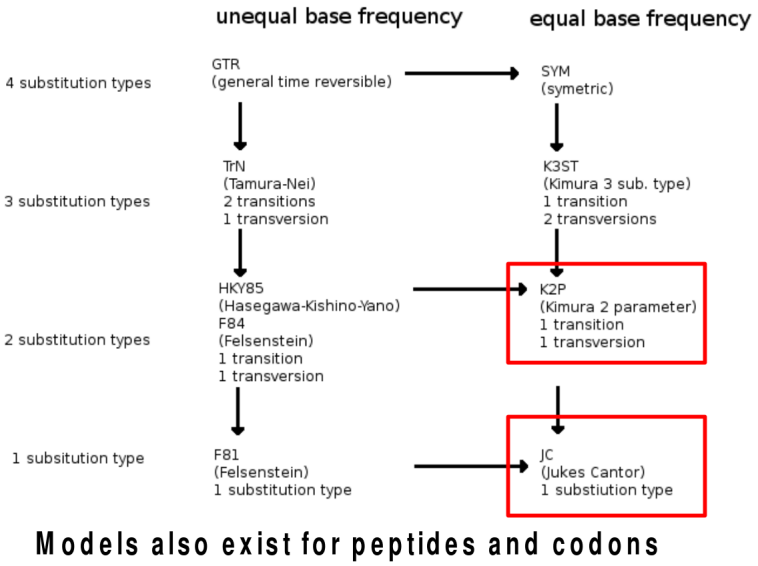


Figure 25.8: Distance models of varying levels of complexity(parameters).

FAQ

Q: Can we use different parameters for different parts of the tree? To account for different mutation rates?

A: Its possible, it is a current area of research.

25.3.2 Distances to Trees

If we have a weighted phylogenetic tree, we can find the total weight (length) of the shortest path between a pair of leaves by summing up the individual branch lengths in the path. Considering all such pairs of leaves, we have a distance matrix representing the data. In distance based methods, the problem is to reconstruct the tree given this distance matrix.

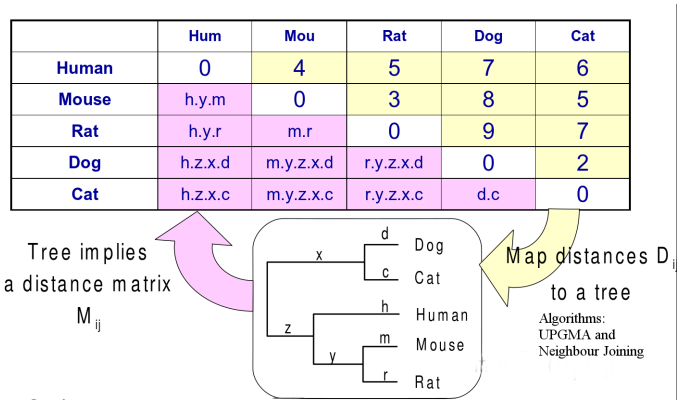


Figure 25.9: Mapping from a tree to a distance matrix and vice versa

FAQ

Q: In Figure 25.9 The m and r sequence divergence metrics can have some overlap so distance between mouse and rat is not simply m+r. Wouldn't that only be the case if there was no overlap?

A: If you model evolution correctly, then you would get evolutionary distance. It's an inequality rather than an equality and we agree that you can't exactly infer that the given distance is the precise distance. Therefore, the sequences' distance between mouse and rat is probably less than m + r because of overlap, convergent evolution, and transversions.

However, note that there is not a one-to-one correspondence between a distance matrix and a weighted tree. Each tree does correspond to one distance matrix, but the opposite is not always true. A distance matrix has to satisfy additional properties in order to correspond to some weighted tree. In fact, there are two models that assume special constraints on the distance matrix:

Ultrametric: For all triplets (a, b, c) of leaves, two pairs among them have equal distance, and the third distance is smaller; i.e. the triplet can be labelled i, j, k such that

$$d_{ij} \leq d_{ik} = d_{jk}$$

Conceptually this is because the two leaves that are more closely related (say i, j) have diverged from the thrid (k) at exactly the same time. and the time separation from the third should be equal, whereas the separation between themselves should be smaller.

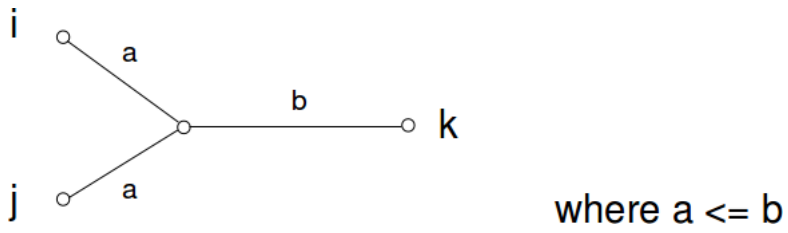


Figure 25.10: Ultrametric distances.

Additive: Additive distance matrices satisfy the property that all quartet of leaves can be labelled i, j, k, l such that

$$d_{ij} + d_{kl} \leq d_{ik} + d_{jl} = d_{il} + d_{jk}$$

This is in fact true for all positive-weight trees. For any 4 leaves in a tree, there can be exactly one topology, i.e.

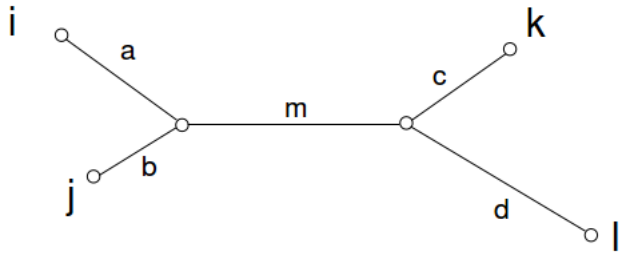


Figure 25.11: Additive distances.

Then the above condition is term by term equivalent to

$$(a + b) + (c + d) \leq (a + m + c) + (b + m + d) = (a + m + d) + (b + m + c)$$

. This equality corresponds to all pairwise distances that are possible from traversing this tree.

These types of redundant equalities must occur while mapping a tree to a distance matrix, because a tree of n nodes has $n - 1$ parameters, one for each branch length, while a distance matrix has n^2 parameters. Hence, a tree is essentially a lower dimensional projection of a higher dimensional space. A corollary of this observation is that not all distance matrices have a corresponding tree, but all trees map to unique distance matrices.

However, real datasets do not exactly satisfy either ultrameric or additive constraints. This can be due to noise (when our parameters for our evolutionary models are not precise), stochasticity and randomness (due to small samples), fluctuations, different rates of mutations, gene conversions and horizontal transfer. Because of this, we need tree-building algorithms that are able to handle noisy distance matrices.

Next, two algorithms that directly rely on these assumptions for tree reconstruction will be discussed.

UPGMA - Unweighted Pair Group Method with Arithmetic Mean

This is exactly same as the method of **Hierarchical clustering** discussed in Lecture 13, Gene Expression Clustering. It forms clusters step by step, from closely related nodes to ones that are further separated. A branching node is formed for each successive level. The algorithm can be described properly by the following steps:

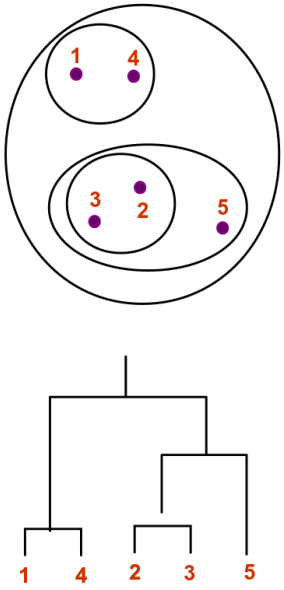


Figure 25.12: UPGMA / Hierarchical Clustering

Initialization:

1. Define one leaf i per sequence x_i.
2. Place each leaf i at height 0.
3. Define Clusters C_i each having one leaf i.

Iteration:

1. Find the pairwise distances d_{ij} between each pairs of clusters C_i, C_j by taking the arithmetic mean of the distances between their member sequences.
2. Find two clusters C_i, C_j such that d_{ij} is minimized.
3. Let $C_k = C_i \cup C_j$.
4. Define node k as parent of nodes i, j and place it at height $d_{ij}/2$ above i, j.
5. Delete C_i, C_j.

Termination: When two clusters C_i, C_j remain, place the root at height $d_{ij}/2$ as parent of the nodes i, j

Weaknesses of UPGMA

Although this method is guaranteed to find the correct tree if the distance matrix obeys the ultrameric property, it turns out to be a inaccurate algorithm in practice. Apart from lack of robustness, it suffers from the molecular clock assumption that the mutation rate over time is constant for all species. However, this is not true as certain species such as rat and mice evolve much faster than others. The following figure illustrates an example where UPGMA fails:

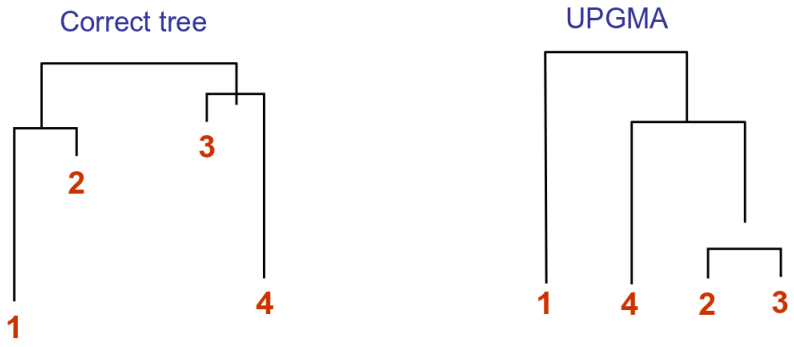


Figure 25.13: UPGMA fails to find the correct tree in this case

Neighbor Joining

The neighbor joining method is guaranteed to produce the correct tree if the distance matrix satisfies the additive property. It may also produce a good tree when there is some noise in the data. The algorithm is described below:

Finding the neighboring leaves: Let

$$D_{ij} = d_{ij} - (r_i + r_j) \text{ where } r_a = \frac{1}{n-2}\sum_k d_{ak},\ a \in \{i, j\}$$

Here n is the number of nodes in the tree; hence, r_i is the average distance of a node to the other nodes. It can be proved that the above modification ensures that D_{ij} is minimal only if i, j are neighbors. (A proof can be found in page 189 of Durbin's book).

Initialization: Define T to be the set of leaf nodes, one per sequence. Let $L = T$

Iteration:

1. Pick i, j such that D_{ij} is minimized.
2. Define a new node k, and set $d_{km} = \frac{1}{2}(d_{im} + d_{jm}d_{ij}) \forall m \in L$
3. Add k to T, with edges of lengths $d_{ik} = \frac{1}{2}(d_{ij} + r_i r_j)$
4. Remove i, j from L
5. Add k to L

Termination: When L consists of two nodes i, j, and the edge between them of length d_{ij}, add the root node as parent of i and j.

25.4 Character-Based Methods

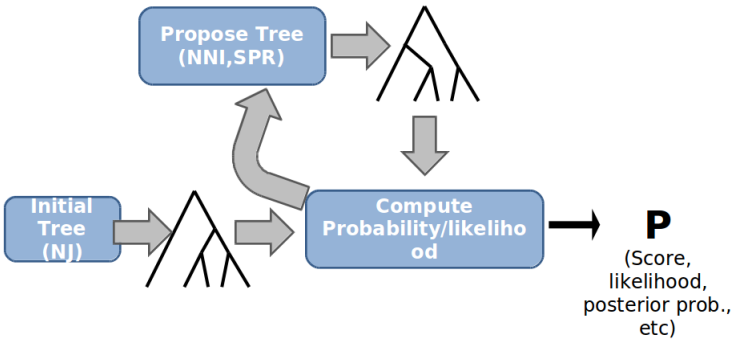


Figure 25.14: An overview of the character based methods

In character-based methods, the goal is to first create a valid algorithm for scoring the probability that a given tree would produce the observed sequences at its leaves, then to search through the space of possible trees for a tree that maximizes that probability. Good algorithms for tree scoring, and while searching the space of trees is theoretically NP-Hard (Due to the large number of possible trees), tractable heuristic search methods can in many cases find good trees. We'll first discuss tree scoring algorithms, then search techniques.

25.4.1 Scoring

There are two main algorithms for tree scoring. The first approach, which we will call parsimony reconstruction, is based on Occam's razor, and scores a topology based on the minimum number of mutations it implies, given the (known) sequences at the leaves. This method is simple, intuitive, and fast. The second approach is a maximum likelihood method which scores trees by explicitly modeling the probability of observing the sequences at the leaves given a tree topology.

Parsimony

Conceptually, this method is simple. It simply assigns a value of for each base pair at each ancestral node such that the number of substitutions is minimized. The score is then just the sum over all base pairs of that minimal number of mutations at each base pair. (Recall that the eventual goal is to find a tree that minimizes that score.)

To reconstruct the ancestral sequences at internal nodes on the tree, the algorithm first scans up from the (known) leaf sequences, assigning a set of bases at each internal node based on its children. Next, it iterates down the tree, picking bases out of the allowed sets at each node, this time based on the node's parents. The following illustrates this algorithm in detail (note that there are $2N-1$ total nodes, indexed from the root, such that the known leaf nodes have indices $N-1$ through $2N-1$):

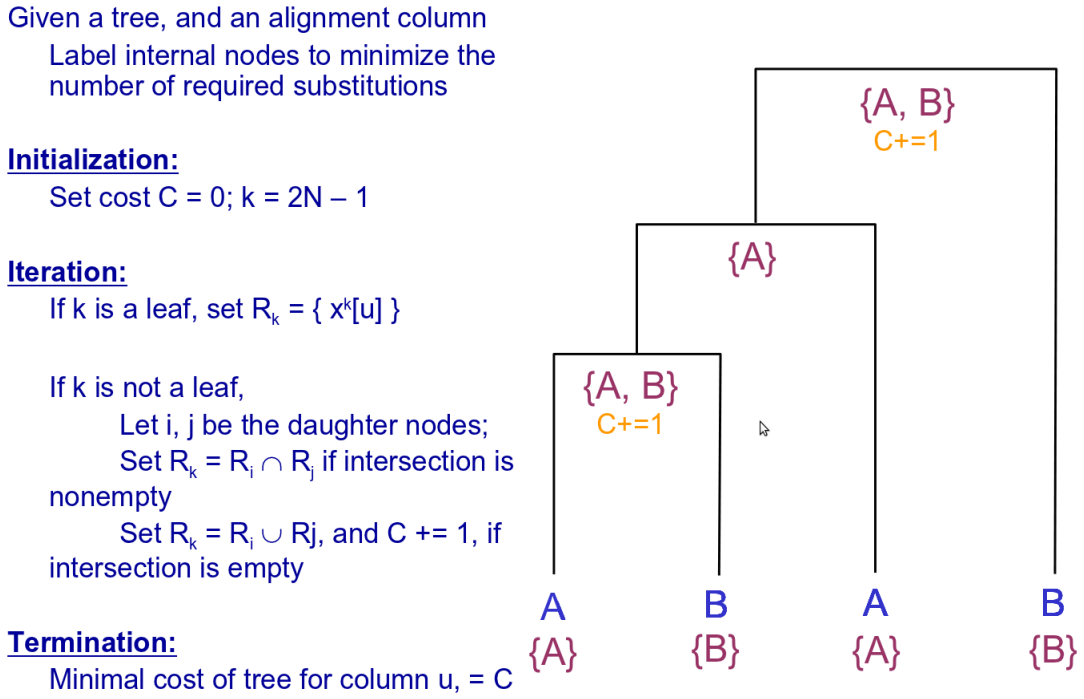


Figure 25.15: Parsimony scoring: union and intersection

Traceback:

1. Choose an arbitrary nucleotide from R_{2N-1} for the root

2. Having chosen nucleotide r for parent k,
   If $r \in R_i$ choose r for daughter i
   Else, choose arbitrary nucleotide from R_i

Easy to see that this traceback produces some assignment of cost C

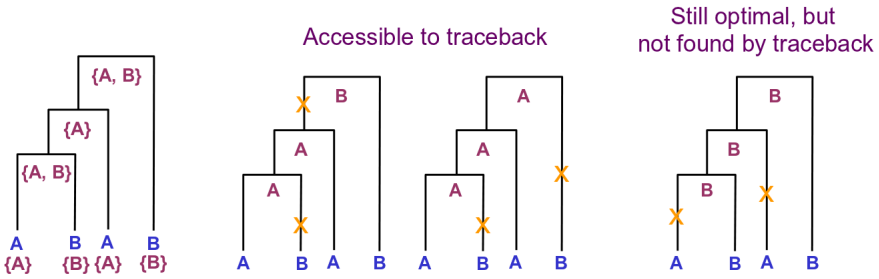


Figure 25.16: Parsimony traceback to find ancestral neucleotides

	M	R	B1	H	B2	D	B3
A	0	1	1	0	1	1	2
C	1	1	2	1	3	1	4
G	1	0	1	1	2	0	2
T	1	1	2	1	3	1	4

- Each cell (N,C) represents the min cost of the subtree rooted at N, if the label at N is C.
- Update table by walking up the tree from the leaves to the root, remembering max choices.
- Traceback from root to leaves

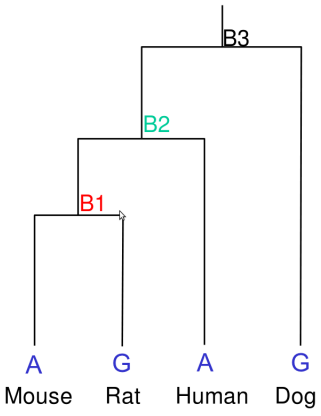


Figure 25.17: Parsimony scoring by dynamic programming

As we mentioned before, this method is simple and fast. However, this simplicity can distort the scores it assigns. For one thing, the algorithm presented here assumes that a given base pair undergoes a substitution along at most one branch from a given node, which may lead it to ignore highly probably internal sequences that violate this assumption. Furthermore, this method does not explicitly model the time represented along each edge, and thus cannot account for the increased chance of a substitution along edges that represent a long temporal duration, or the possibility of different mutation rates across the tree. Maximum likelihood methods largely resolve these shortcomings, and are thus more commonly used for tree scoring.

Maximum Likelihood - Peeling Algorithm

As with the general Maximum likelihood methods, this algorithm scores a tree according to the (log) joint probability of observing the data and the given tree, i.e. $P(D, T)$. The peeling algorithm again considers individual base pairs and assumes that all sites evolve independently. As in the parsimony method, this algorithm considers all base pairs independently: it calculates the probability of observing the given characters at each base pair in the leaf nodes, given the tree, a set of branch lengths, and the maximum likelihood assignment of the internal sequence, then simply multiplies this probabilities over all base pairs to get the total probability of observing the tree. Note that the explicit modeling of branch lengths is a difference from the previous approach.

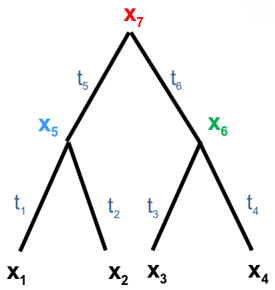


Figure 25.18: A tree to be scored using the peeling algorithm. n=4

Here each node has a character x_i and t_i is the corresponding branch length from its parent. Note that we already know the values $x_1, x_2 \cdots x_n$, so they are constants, but $x_{n+1}, \cdots x_{2n-1}$ are unknown characters at ancestral nodes which are variables to which we will assign maximum likelihood values. (Also note that we have adopted a leaves-to-root indexing scheme for the nodes, the opposite of the scheme we used before.) We want to compute $P(x_1 x_2 \cdots x_n|T)$. For this we sum over all possible combinations of values at the ancestral nodes. this is called marginalization. In this particular example

$$P(x_1x_2x_3x_4|T) = \sum_{x_5}\sum_{x_6}\sum_{x_7} P(x_1x_2\cdots x_7|T)$$

There are 4^{n-1} terms in here, but we can use the following factorization trick:

$$= \sum_{x_5}\sum_{x_6}\sum_{x_7} P(x_1|x_5,t_1)P(x_2|x_5,t_2)P(x_3|x_6,t_3)P(x_4|x_6,t_4)P(x_5|x_7,t_5)P(x_6|x_7,t_6)P(x_7)$$

Here we assume that each branch evolves independently. And the probability $P(b|c,t)$ denotes the probability of base c mutating to base b given time t, which is essentially obtained from the Jukes Cantor model or some more advanced model discussed earlier. Next we can move the factors that are independent of the summation variable outside the summation. That gives:

$$= \sum_{x_7}\left[P(x_7)\left(\sum_{x_5} P(x_5|x_7,t_5)P(x_1|x_5,t_1)P(x_2|x_5,t_2)\right)\left(\sum_{x_6} P(x_6|x_7,t_6)P(x_3|x_6,t_3)P(x_4|x_6,t_4)\right)\right]$$

Let T_i be the subtree below i. In this case, our $2n-1 \times 4$ dynamic programming array computes $L[i,b]$, the probability $P(T_i|x_i = b)$ of observing T_i, if node i contains base b. Then we want to compute the probability of observing $T = T_{2n-1}$, which is

$$\sum_b P(x_{2n-1} = b)L[2n-1, b]$$

Note that for each ancestral node i and its childer j, k, we have

$$L[i,b] = \left(\sum_c P(c|b,t_j)L[j,c]\right)\left(\sum_c P(c|b,t_k)L[k,c]\right)$$

Subject to the initial conditions for the leaf nodes, i.e. for $i \leq n$:

$$L[i,b] = 1 \text{ if } x_i = b \text{ and } 0 \text{ otherwise}$$

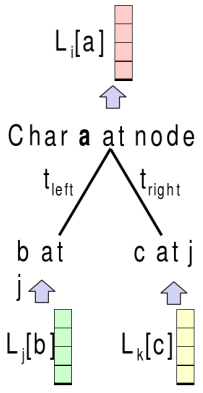
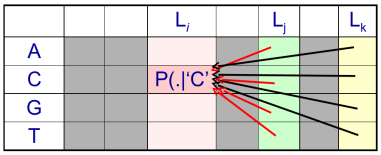
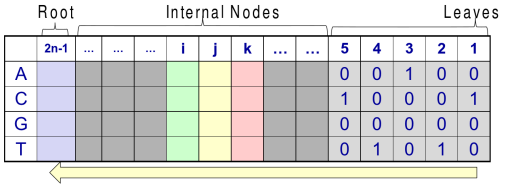


Figure 25.19: The recurrence



(a) Filling the matrix



(b) Inititalization and direction

Note that we still do not have the values $P(x_{2n-1} = b)$. It is usually assigned equally or from some prior distribution, but it does not affect the results greatly. The final step is of course to multiply all the probabilities for individual sites to obtain the probability of observing the set of entire sequences. In addition, once we have assigned the maximum likelihood values for each internal node given the tree structure and the set of branch lengths, we can multiply the resulting score by some prior probabilities of the tree structure and the set of branch lengths, which are often generated using explicit modeling of evolutionary processes, such as the Yule process or birth-death models like the Moran process. The result of this final multiplication is called the a posteriori probability, using the language of Bayesian inference. The overall complexity of this algorithm is $O(nmk^2)$ where n is the number of leaves (taxa), m is the sequence length, and k is the number of characters.

There are addvantages and disadvantages of this algorithm. Such as

Advantages:

1. Inherently statistical and evolutionary model-based.
2. Usually the most consistent of the methods available.
3. Used for both character and rate analyses
4. Can be used to infer the sequences of the extinct ancestors.
5. Account for branch-length effects in unbalanced trees.
6. Nucleotide or amino acid sequences, other types of data.

Disadvantages:

1. Not as simple and intuitive as many other methods.
2. Computationally intense Limited by, number of taxa and sequence length).
3. Like parsimony, can be fooled by high levels of homoplasy.
4. Violations of model assumptions can lead to incorrect trees.

25.4.2 Search

A comprehensive search over the space of all trees would be extremely costly. The number of full rooted trees with $n + 1$ leaves is the n-th catalan number

$$C_n = \frac{1}{n+1}\binom{2n}{n} \approx \frac{4^n}{n^{3/2}\sqrt{\pi}}$$

Moreover, we must compute the maximum likelihood set of branch lengths for each of these trees. Thus, it is an NP-Hard problem to maximize the score absolutely for all trees. Fortunately, heuristic search algorithms can generally identify good solutions in the tree space. The general framework for such search algorithms is as follows:

Inititalization: Take some tree as the base of iteration (randomly or according to some other prior, or from the distance based direct algorithms).

Proposal: Propose a new tree by randomly modifying the current tree slightly.

Score: Score the new proposal according to the methods described above.

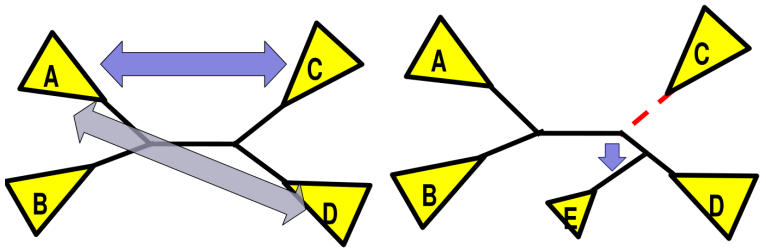
Select: Randomly select the new tree or the old tree (corresponding probabilities according to the score(likelihood) ratio.

Iterate: Repeat to proposal step unless some termination criteria is met (some threshold score or number of steps reached.

the basic idea here is the heuristic assumption that the scores of closely related trees are similar, so that good solutions may be obtained by successive local optimization, which is expected to converge towards a overall good solution.

Tree Proposal

One method for modifying trees is the Nearest Neighbor Exchange (NNI), illustrated below.



(c) Before (d) After

Figure 25.20: An unit step using Nearest Neighbor Interchange scheme

Another common method, not described here, is Tree Bisection and Join (TBJ). The important criteria for such proposal rules is that:

(a) The tree space should be connected, i.e. any pair of trees should be obtainable from each other by successive proposals.

(b) An individual new proposal should be sufficiently close to the original. So that it is more likely to be a good solution by virtue of the proximity to an already discovered good solution. If individual steps are too big, the algorithm may move away from an already discovered solution (also depends on the selection step). In particular, note that the measure of similarity by which the measure these step sizes is precisely the difference in the likelihood scores assigned to the two trees.

Selection

Choosing whether or not to adopt a given proposal, like the process of generating the proposal itself, is inherently heuristic and varies. A general rules of thumb is:

1. If the new one has a better score, always accept it.
2. If it has a worse score, there should be some probability of selecting it, otherwise the algorithm will soon fixate in a local minima, ignoring better alternatives a little far away.

3. There should not be too much probability of selecting an worse new proposal, otherwise, it risks rejecting a known good solution.

It is the trade-off between the steps 2 and 3 that determines a good selection rule. Metropolis Hastings is a Markov Chain Monte Carlo Method (MCMC) that defines specific rules for exploring the state space in a way that makes it a sample from the posterior distribution. These algorithms work somewhat well in practice, but there is no guarantee for finding the appropriate tree. So a method known as bootstrapping is used, which is basically running the algorithm over and over using subsets of the base pairs in the leaf sequences,. then favoring global trees that match the topologies generated by using only these subsequences.

25.5 Possible Theoretical and Practical Issues with Discussed Approach

A special point must be made about distances. Since distances are typically calculated between aligned gene sequences, most current tree reconstruction methods rely on heavily conserved genes, as non-conserved genes would not give information on species without those genes. This causes the ignoring of otherwise useful data. Therefore, there are some algorithms that try to take into account less conserved genes in reconstructing trees but these algorithms tend to take a long time due to the NP-Hard nature of reconstructing trees.

Additionally, aligned sequences are still not explicit in regards to the events that created them. That is, combinations of speciation, duplication, loss, and horizontal gene transfer (hgt) events are easy to mix up because only current DNA sequences are available. (see [9] for a commentary on such theoretical issues) A duplication followed by a loss would be very hard to detect. Additionally, a duplication followed by a speciation could look like an HGT event. Even the probabilities of events happening is still contested, especially horizontal gene transfer events.

Another issue is that often multiple marker sequences are concatenated and the concatenated sequence is used to calculate distance and create trees. However, this approach assumes that all the concatenated genes had the same history and there is debate over if this is a valid approach given that events such as hgt and duplications as described above could have occurred differently for different genes. [8] is an article showing how different phylogenetic relationships were found depending on if the tree was created using multiple genes concatenated together or if it was created using each of the individual genes. Conversely, additional [4] claims that while hgt is prevalent, orthologs used for phylogenetic reconstruction are consistent with a single tree of life. These two issues indicate that there is clearly debate in the field on a non arbitrary way to define species and to infer phylogenetic relationships to recreate the tree of life.

25.6 Towards final project

25.6.1 Project Ideas

1. Creating better distance models such as taking into account duplicate genes or loss of genes. It may also be possible to analyze sequences for peptide coding regions and calculate distances based on peptide chains too.
2. Creating a faster/more accurate search algorithm for turning distances into trees.
3. Analyze sequences to calculate probabilities of speciation, duplication, loss, and horizontal gene transfer events.
4. Extending an algorithm that looks for HGTs to look for extinct species. A possible use for HGTs is that if a program were to infer HGTs between different times, it could mean that there was a speciation where one branch is now extinct (or not yet discovered) and that branch had caused an HGT to the other extant branch.

25.6.2 Project Datasets

1. 1000 Genomes Project http://www.1000genomes.org/
2. Microbes Online http://microbesonline.org/

25.7 What Have We Learned?

In this chapter, we have learnt different methods and approaches for reconstructing Phylogenetic trees from sequence data. In the next chapter, its application in gene trees and species trees and the relationship between those two will be discussed, as well as modelling phylogenies among populations within a species and between closely related species.

Bibliography

[1] 1000 genomes project.

[2] et al Ciccarelli, Francesca. Toward automatic reconstruction of a highly resolved tree of life. *Science*, 311, 2006.

[3] Tal Dagan and William Martin. The tree of one percent. *Genome Biology*, Nov 2006.

[4] Ochman Howard Daubin Vincent, Moran Nancy A. Phylogenetics and the cohesion of bacterial genomes. *Science*, 301, 2003.

[5] A.J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein familes. *Nucleic Acids Research*, 30(7):1575–1584, Apr 2002.

[6] Stephanie Guindon and Olivier Gascuel. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systems Biology*, 52(5):696–704, 2003.

[7] Sanderson MJ. r8s: Inferring absolute rates of molecular evolution and divergence times in the absence of a molecular clock. *Bioinformatics*, 19(2):301–302, Jan 2003.

[8] R. Thane Papke, Olga Zhaxybayeva, Edward J Fiel, Katrin Sommerfeld, Denise Muise, and W. Ford Doolittle. Searching for species in haloarchaea. *PNAS*, 104(35):14092–14097, 2007.

[9] Douglas L Theobald. A formal test of the theory of universal common ancestry. *Nature*, 465:219–222, 2010.

CHAPTER

TWENTYSIX

PHYLOGENOMICS II

Guest Lecture by
Matt Rasmussen
2012: Updated by Orit Giguzinsky and Ethan Sherbondy

Figures

26.1 Introduction

In the previous chapter, we covered techniques for reasoning about evolution in terms of trees of descent. The algorithms we covered for tree-building, UPGMA and neighbor-joining, assumed that we were comparing fully aligned sections of sequences.

In this section, we present additional models for using phylogenetic trees in different contexts. Here we clarify the differences between species and gene trees. We then cover a framework called reconciliation which lets us effectively combine the two by mapping gene trees onto species trees. This mapping gives us a means of inferring gene duplication and loss events.

We will also present a phylogenetic perspective for reasoning about population genetics. Since population genetics deals with relatively recent mutation events, we offer the Wright-Fisher model as a tool for representing changes in whole populations. Unfortunately, when dealing with real-world data, we usually are only able to sequence genes from the current living descendants of a group. As a remedy to this shortcoming, we cover the Coalescent model, which you can think of as a time-reversed Wright-Fisher analog.

By using coalescence, we gain a new means for estimating divergence times and population sizes across multiple species. At the end of the chapter, we touch briefly on the challenges of using trees to model recombination events and summarize recent work in the field along with frontiers open for exploration.

26.2 Inferring Orthologs/Paralogs, Gene Duplication and Loss

There are two commonly used trees, Species tree and Gene tree. This section explains how these trees can be used and how to fit a gene tree inside a species tree (reconciliation).

26.2.1 Species Tree

Species trees that show how different species evolved from one another. These trees are created using morphological characters, fossil evidence, etc. The leaves of each tree are labeled as species and the rest of the tree shows how these species are related. An example of a species tree is shown in Figure 26.1.

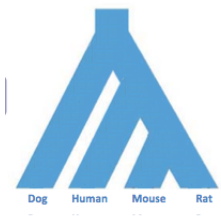


Figure 26.1: Species Tree

26.2.2 Gene Tree

Gene trees are trees that look at specific genes in different species (leaves are genes). The leaves of gene trees are labeled with gene sequences or gene ids associated with specific sequences. Figure 26.2 shows an example of a gene tree that has 4 genes (leaves). The sequences associated with each gene are presented on the right side of Figure 26.2.

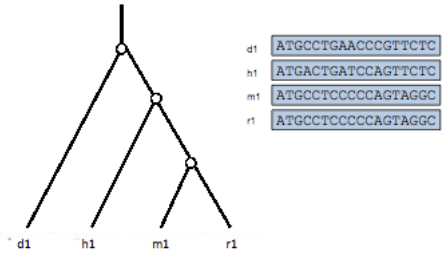


Figure 26.2: Gene Tree

26.2.3 Gene Family Evolution

Gene trees evolve inside a species tree. An example of a gene tree contained in a species tree is shown in Figure 26.3 below.

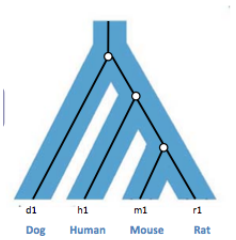


Figure 26.3: Gene Tree Inside a Species Tree

The next sub section explains how we can fit gene trees inside a species trees using Reconciliation.

26.2.4 Reconciliation

Reconciliation is an algorithm that helps compare gene trees to genome trees by fitting a gene tree fits inside a species tree. This is done by by mapping the vertices in the gene tree to vertices in the species tree. This sub section will focus on Reconciliation, related definitions, algorithm (Maximum Parsimony Reconciliation algorithm) and examples.

Definitions

Two genes are **orthologs** if their recent common ancestor (MRCA) is a speciation (splitting into different species).
Paralogs are genes whose MRCA is a duplication.
Figure 26.4 below illustrates how these types of genes can be represented in a gene tree. The tree below has 4 speciation nodes, one duplication and one loss.

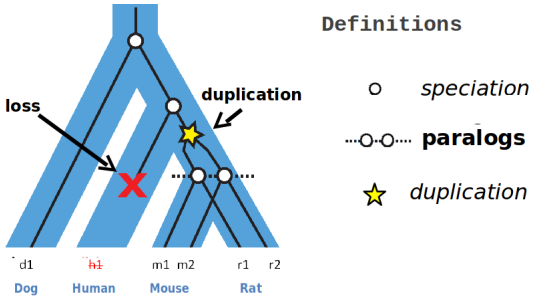


Figure 26.4: Gene Family Evolution: Gene Trees and Species Trees

A mapping diagram is a diagram that shows the node mapping from the gene tree to the species tree. Figure 26.5 shows an example of a mapping diagram.

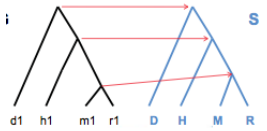


Figure 26.5: Mapping Diagram

A nesting diagram shows how the gene tree can be nested inside the species tree. For every mapping diagram there is a nesting diagram. Figure 26.6 shows an example of a possible nesting diagram for the mapping diagram in Figure 26.5.

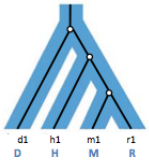


Figure 26.6: Nesting Diagram

Maximum Parsimony Reconciliation (MPR) algorithm

MPR is an algorithm that fits a gene tree in a species tree while minimizing the number of duplications and deletions.

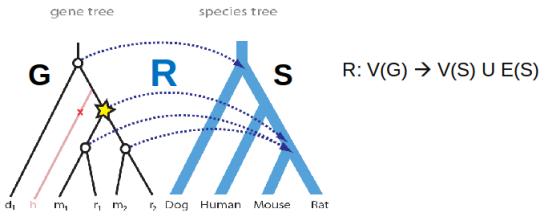


Figure 26.7: Maximum Parsimony Reconciliation (MPR)

Given a gene tree and a species tree, the algorithm finds the reconciliation that minimizes the number of duplications and deletions. Figure 26.7 above shows an example of a possible mapping from a gene tree to a species tree. Figure 26.8 presents the pseudocode for the MPR algorithm.

Solve recursively:
•R[v] = *species of* v if v ∈ L(G)
•R[v] = LCA(R[right(v)] , R[left(v)]) if v ∈ I(G)
  •LCA = "least common ancestor"
    (also called "most recent common ancestor")

Labeling events:
•v is a dup if R[v] = R[right(v)] or R[left(v)]
•Branch above v has at least one loss if R[parent(v)] != R[v] or parent[R[v]]

Figure 26.8: Maximum Parsimony Reconciliation Recursive Algorithm

We map the arrows low as possible, since lower mapping usually results in fewer events. However, we cannot map too low. We map as low as we can without violating the descendent-ancestor relationships. The algorithm goes recursively from bottom up, starting from the leaves. We already know the mapping for the leaves, so we can easily map them. To map the ancestors, for each node (going recusively up the tree) we look at the right child and left child and take the least common ancestor (LCA) of the species that they map to. If a node maps to its right or left child, we know there is a duplication. An expected branch that does not exist indicates a loss.

Reconciliation Examples

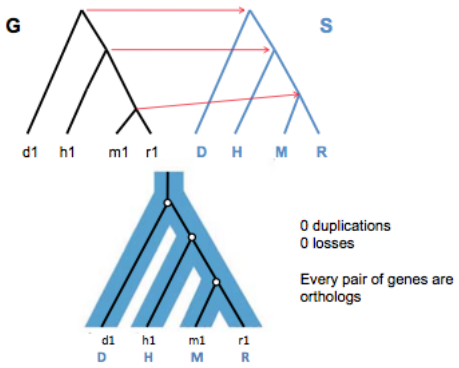


Figure 26.9: Reconciliation Example 1, simple mapping case

In Figure 26.9, the nodes can be mapped straight across, since there are no duplications or losses.

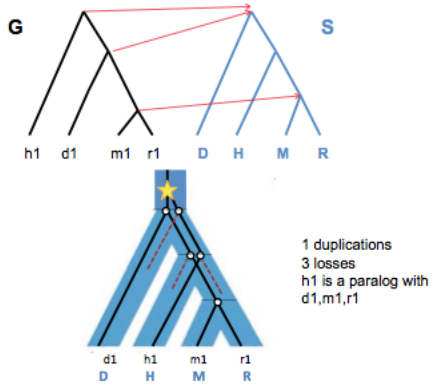


Figure 26.10: Reconciliation Example 2, parsimonious reconciliation for complex case

In Figure 26.10, we see a parsimonious (minimum number of losses and duplications) reconciliation for a case in which nodes from the gene tree cannot be mapped straight across.

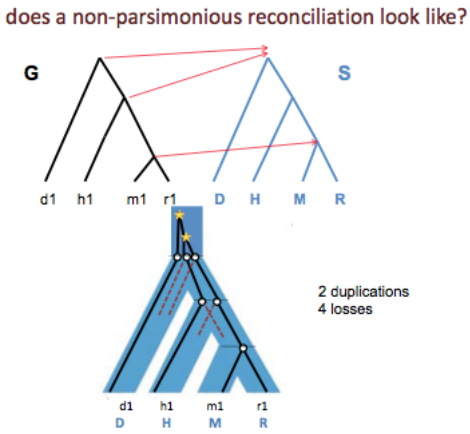


Figure 26.11: Reconciliation Example 3, non parsimonious reconciliation for complex case

Figure 26.11 shows a non-parsimonious reconciliation . The parsimonious mapping for the same trees is shown in Figure 26.9.

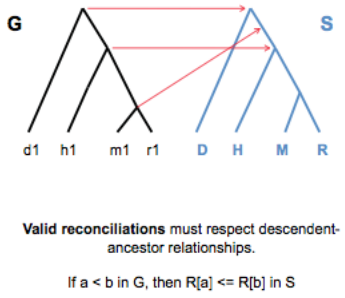


Figure 26.12: Reconciliation Example 4, invalid Reconciliation

Figure 26.12 shows an invalid reconciliation. This reconciliation is invalid since it does not respect descendent-ancestor relationships. In order for this reconciliation to be possible, the descendent would have to travel back in time and be created before its ancestor. Clearly, such a scenario would be impossible. A valid reconciliation must satisfy the following: **If a $<$ b in G, then R[a] $\leq$ R[b] in S**.

26.3 Reconstruction

In the previous section we learned how to compare gene trees and species trees. In this section, we will use this information to reconstruct gene trees and species trees.

26.3.1 Species Tree Reconstruction

In the past, it was really hard to identify a marker gene for a specific species. As sequencing improved we started having lots of sequencing data, people started building trees for different loci. The tree you got highly dependent on the tree you used. Possible reasons why trees differ include noise (from statistical estimate errors and noise), hidden duplications and losses and allele sorting in a population.

Species Tree Reconstruction Problem

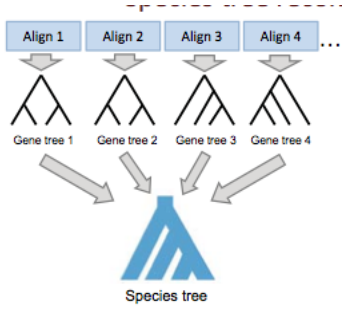


Figure 26.13: Species Tree Reconstruction

Given lots of different gene trees that disagree, our goal is to make them into once species tree (as shown in Figure 26.13. There are lots of different algorithms that reconstruct species trees. These algorithms include Supermatrix methods (Rokas 2003, Ciccareli 2006), Supertree methods (Creevey & McInerney 2005), Minimizing Deep Coalescence (Maddison & Knowles 2006) and Modeling coalescence (Liu & Pearl 2007).

One way to do this, which is mostly effective for noisy data, is to pull more data together in order to increase accuracy. This is done by concatenating gene alignments into a super-matrix.

Another method involves building a tree for each one and using a consensus method to summarize these trees. Then we identify branches that frequently across the trees and build a species tree that has the branches that occur most frequently.

There is another way to reconstruct a species tree, which is effective in case the gene trees disagree because of duplications and losses. The goal is to find the species tree that applies the fewest duplications.

We build all the gene trees and then propose a species tree. Next, we use reconciliation to determine the number of events each gene tree combined with the proposed species tree implies. Then, we propose other species trees and move branches around. Wrong species trees tend to have lots of events that did not happen. The correct tree should have the fewest number of events.

26.3.2 Improving Gene Tree Reconstruction and Learning Across Gene Trees

We can use methods similar to those described above to build better gene trees. This can be done by using information from a species tree to study a gene tree of interest. For example, species trees can be used to determine when losses and duplications occurred. The idea is that we can use the fact that species trees are often built from the entire genome, to obtain more information about related gene trees. We can use both the branch length and the number of events to do this.

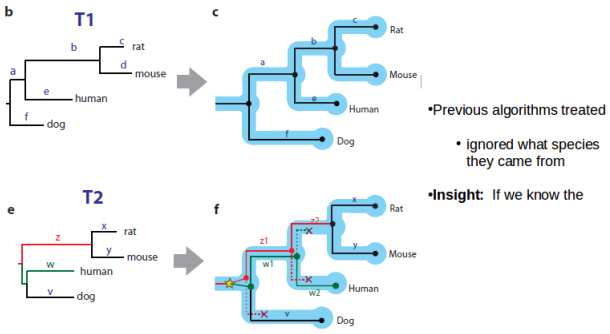


Figure 26.14: Using species trees to improve gene tree reconstruction.

If we know the species tree, we can develop a model for what kind of branch lengths we can expect. We can use conserved gene order to tell orthologs and build trees.

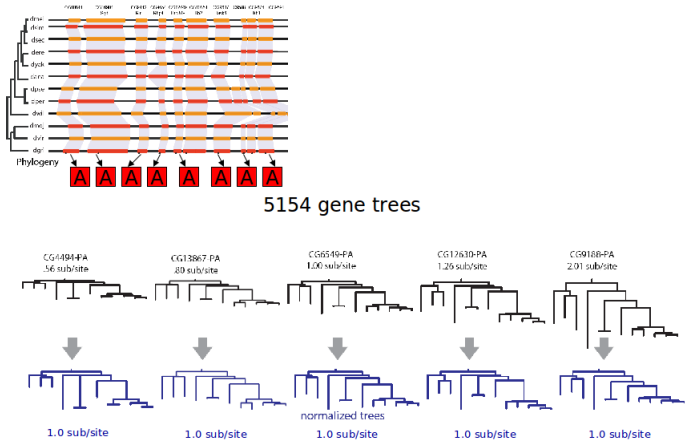


Figure 26.15: We can develop a model for what kind of branch lengths we can expect. We can use conserved gene order to tell orthologs and build trees.

When a gene is fast evolving in one species, it is fast evolving in all species. We can model a branch length as two different rate components. One is gene specific(present across all species) and the other is species specific, which is customized to a specific species.

This method greatly improves reconstruction accuracy.

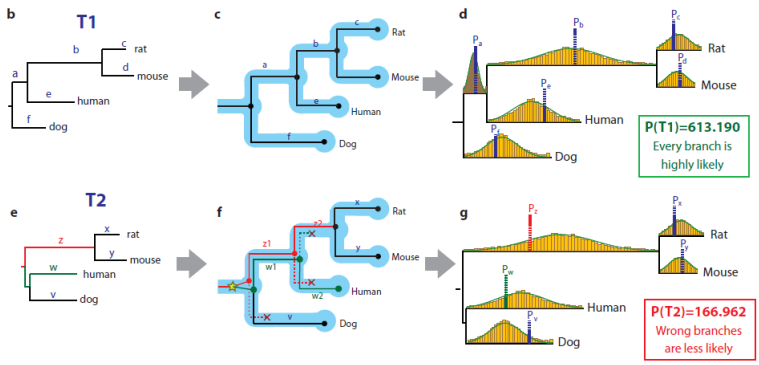


Figure 26.16: Branch length can be modeled as two different rate components: gene specific and species specific.

26.4 Modeling Population and Allele Frequencies

With the advent of next-gen sequencing, it is becoming economical to sequence the genomes of many individuals within a population. In order to make sense of how alleles spread through a population, it's helpful to have a model to compare data against. The **Wright-Fisher** reproduction model has filled this role for the past 70 years.

26.4.1 The Wright-Fisher Model

Like HMMs, Wright-Fisher is a Markov process: at each step, the system randomly progresses, and the current state of the system depends only on the previous state. In this case, state transitions represent reproduction. By modeling the transmission of chromosomes to offspring, we can study genetic drift.

The model makes a number of simplifying assumptions:

1. Population size, **N**, is constant at each generation.
2. Only members of the same generation reproduce (no overlap).
3. Reproduction occurs at random.
4. The gene being modeled only has 2 alleles.
5. Genes undergo neutral selection.

Note that Wright-Fisher is not an appropriate choice if you're trying to model the change in frequency of a gene that is positively or negatively selected for. If we use Wright-Fisher to model the chromosomes of diploid individuals, the population size of the model becomes **2N**.

In English, here's how Wright-Fisher works:

At every generation, for each child, we randomly select from the parents (with replaccement). The allele of the child becomes that of the randomly selected parent.

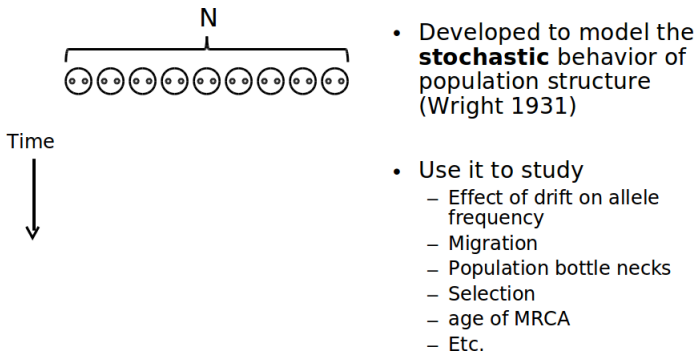


Figure 26.17: The Wright-Fisher model

We repeat this process for many generation, with the children serving as the new parents, ignoring the ordering of chromosomes.

It really is that simple. To determine the probability of k copies of an allele existing in the child generation when it had a frequency of p in the parent generation, we can use this formula:

$$\binom{2N}{k} p^k q^{2N-k} \tag{26.1}$$

Here, $q = (1 - p)$. It is the frequency of non-p alleles in the parent generation.

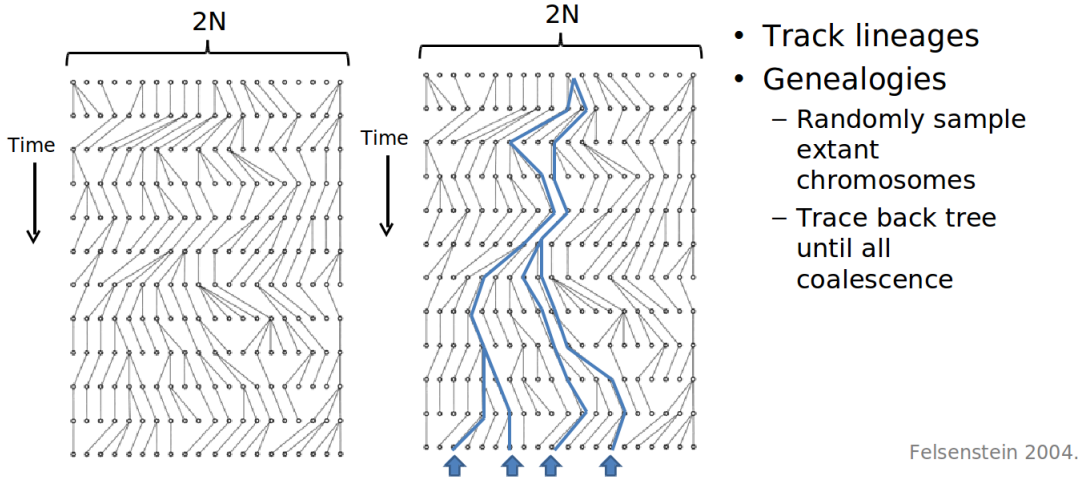


Figure 26.18: Many iterations of Wright-Fisher yielding a lineage tree

Now we can begin to explore such questions as: how probable is it and how many generations is it expected to take for a given allele to become **fixed**, meaning the allele is present in *every* member of the population?

The expected time (in generations) for fixation, given the assumptions made by Wright-Fisher, is proportionate to $4N_E$, where N_E is the effective population size.

Again, it's important to keep in mind the limitations of this model and ask if it actually makes sense for the system you're trying to represent. Consider how you could tweak the proposed model to account for a selection coefficient ranging between -1 (lethal negative selection) and 1 (strong positive selection).

26.4.2 The Coalescent Model

The problem with the Wright-Fisher model is that it assumes you know the allele frequencies of the ancentral generation. When dealing with the genomes of present species, these quantities are unknown. The Coalescent Model solves this conundrum by thinking retrospectively. That is to say: we start with the alleles of the *current* generation, and work our way *backwards* in time. The basic Coalescence Model makes the same assumptions as Wright-Fisher. At each generation, we ask: what is the probability of the two identical alleles coalescing, or sharing a parent, in the previous generation.

We can pose the probability of a coalescence event occuring in the previous generation as the probability of coalescence *not* occuring in any of the $t-1$ generations prior to the last one, times the probability of it occuring in the previous (the t-th) generation. This is equivalent to the expression:

$$P_c(t) = \left(1 - \frac{1}{2N_e}\right)^{t-1} \left(\frac{1}{2N_e}\right) \tag{26.2}$$

Where N_e is the effective population size.

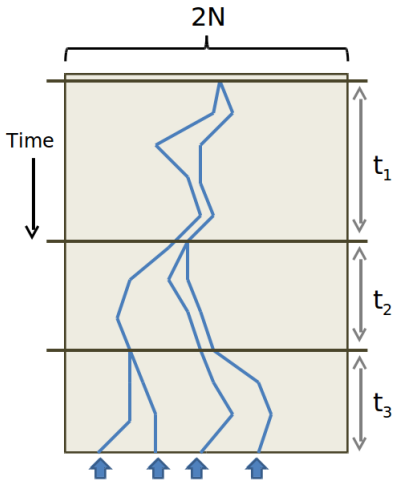


Figure 26.19: The coalescent model.

By approximating this geometric distribution as an exponential one: $P_c(t) = \frac{1}{2N_e} e^{-\left(\frac{t-1}{2N_e}\right)}$, we can determine the expected number of generations back until coalescence, which turns out to be $2N_e$, with a standard deviation of $2N_e$.

To ask about the coalescence of *multiple* lineages at a given generation, we must, as in Wright-Fisher, use a binomial distribution. The probability of k lineages coalescing for the *first* time at generation t is:

$$P(T_k = t) = \left(1 - \binom{k}{2}\frac{1}{2N}\right)^{t-1}\binom{k}{2}\frac{1}{2N} \tag{26.3}$$

And again, this can be approximated with an exponential distribution for sufficiently large k. The individual at which two lineages converge is referred to as the **Most Recent Common Ancestor**. By continually moving backwards until all ancestors coalesce, we end up with a new kind of tree! And by comparing the tree resulting from coalescence with a gene tree we've constructed, discrepancies between the two may signal that certain assumptions of the Coalescent Model have been violated. Namely, selection may be occuring.

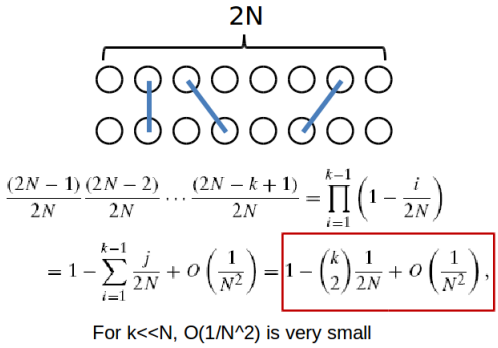


Figure 26.20: Geometric probability distribution for coalescent events in k lineages.

26.4.3 The Multispecies Coalescent Model

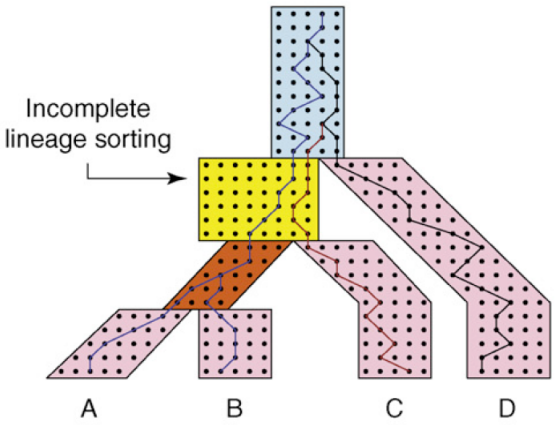


Figure 26.21: Multispecies Coalescent Model.

We can take this idea once step further and track coalescence events across multiple species. Here, each genome of an individual species is treated as a lineage.

Note that there is a lag time between the separation of two populations and the time at which two gene lineages coalesce into a common ancestor. Also note how the rate of coalescence slows down as N gets bigger and for short branches.

In the image above, deep coalescence is depicted in light blue for three lineages. The species and gene trees here are incongruent since C and D are sisters in gene tree but not the species tree.

There is a $\frac{2}{3}$ chance that incongruence will occur because once we get to the light blue section, Wright-Fisher is memoryless and there is only $\frac{1}{3}$ chance that it will be congruent. The effect of incongruence is called **Incomplete Lineage Sorting**. By measuring the frequency at which **ILS** occurs, we gain insight into unusually large populations or unsually short branch lengths within the species tree.

You can build a maximum parsimony species tree based on the notion of minimizing the number of ILS events rather than minimizing implied duplication/loss events as covered previously. It is even possible to combine these two methods to, ideally, create a phylogeny that is more accurate than either of them would be individually.

26.5 SPIDIR

26.5.1 Background

As presented in the supplementary information for SPIDIR, a gene family is the set of genes that are descendents of a single gene in the most recent common ancestor (MRCA) of all species under consideration. Furthermore, genetic sequences undergo evolution at multiple scales, namely at the level of base pairs, and at the level of genes. In the context of this lecture, two genes are orthologs if their MRCA is a speciation event; two genes are paralogs if their MRCA is a duplication event.

In the genomic era, the species of a modern genes is often known; ancestral genes can be inferred by reconciling gene- and species-trees. A reconciliation maps every gene-tree node to a species-tree node. A common technique is to perform Maximum Parsimony Reconciliation (MPR), which finds the reconciliation R implying the fewest number of duplications or losses using the recursion over inner nodes v of a gene tree G. MPR fist maps each leaf of the gene tree to the corresponding species leaf of the species tree. Then the internal nodes of G are mapped recursively:

$$R(v) = MRCA(R(right(v)), R(left(v)))$$

If a speciation event and its ancestral node are mapped to the same node on the species tree. Then the ancestral node must be an duplication event.

Using MPR, the accuracy of the gene tree is crucial. Suboptimal gene trees may lead to an excess of loss and duplication events. For example, if just one branch is misplaced (as in **??**) then reconciliation infers 3 losses and 1 duplication event. In [6], the authors show that the contemporaneous current gene tree methods perform poorly (60% accuracy) on single genes. But if we have longer concatenated genes, then accuracy may go up towards 100%. Furthermore, very quickly or slowly evolving genes carry less information as compared with moderately diverging sequences (40-50% sequence identity), and perform correspondingly worse. As corroborated by simulations, single genes lack sufficient information to reproduce the correct species tree. Average genes are too short and contains too few phylogenetically informative characters. While many early gene tree construction algorithms ignored species information, algorithms like SPIDIR capitalize on the insight that the species tree can provide additional information which can be leveraged for gene tree construction. Synteny can be used to independently test the relative accuracy of different gene tree reconstructions. This is because syntenic blocks are regions of the genome where recently diverged organisms have the same gene order, and contain much more information than single genes.

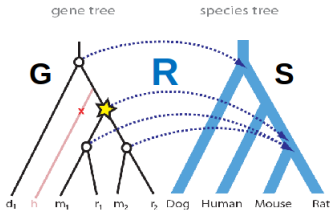


Figure 26.22: MPR reconciliation of genes and species tree.

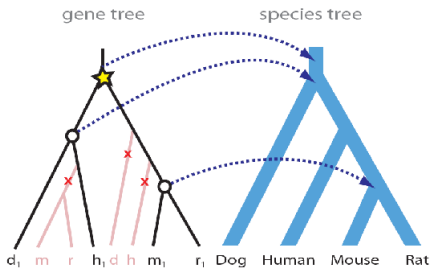


Figure 26.23: Inaccuracies in gene tree.

There have been a number of recent phylogenomic algorithms including: RIO [2], which uses neighbor joining (NJ) and bootstrapping to deal with incogruencies, Orthostrapper [7], which uses NJ and reconciles to a vague species tree, TreeFAM [3], which uses human curation of gene trees as well as many others. A number of algorithms take a more similar track to SPIDIR [6], including [4], a probabilistic reconciliation algorithm [8], a Bayesian method with a clock,[9],and parsimony method using species tree , as well as more

recent developments: [1] a Bayesian method with relaxed clock and [5], a Bayesian method with gene and species specific relaxed rates (an extension to SPIDIR) .

26.5.2 Method and Model

SPIDIR exemplifies an iterative algorithm for gene tree construction using the species tree. In SPIDIR, the authors define a generative model for gene-tree evolution. This consists of a prior for gene-tree topology and branch lengths. SPIDIR uses a birth and death process to model duplications and losses (which informs the prior on topology) and then then learns gene-specific and species-specific substitution rates (which inform the prior on branch lengths). SPIDIR is a *Maximum a posteriori (MAP)* method, and, as such, enjoys several nice optimality criteria.

In terms of the estimation problem, the full SPIDIR model appears as follows:

$$argmax L,T,RP(L,T,R|D,S,\Theta) = argmax L,T,RP(D|T,L)P(L|T,R,S,\Theta)P(T,R|S,\Theta)$$

The parameters in the above equation are: D = alignment data , L = branch length T = gene tree topology , R = reconciliation , S = species tree (expressed in times) , Θ = (gene and species specific parameters [estimated using EM training], λ, μ dup/loss parameters)). This model can be understood through the three terms in the right hand expression, namely:

1. the sequence model– $P(D|T,L)$. The authors used the common HKY model for sequence substitutions, which unifies Kimura's two parameter model for transitions and transversions with Felsenstein's model where substitution rate depends upon nucleotide equilibrium frequency.

2. the first prior term, for the rates model– $P(L|T,R,S,\Theta)$, which the authors compute numerically after learning species and gene specific rates.

3. the second prior term, for the duplication/loss model– $P(T,R|S,\Theta)$, which the authors describe using a birth and death process.

Having a rates model is very rates model very useful, since mutation rates are quite variable across genes. In the lecture, we saw how rates were well described by a decomposition into gene and species specific rates. In lecture we saw that an inverse gamma distribution appears to parametrize the gene specific substitution rates, and we were told that a gamma distribution apparently captures species specific substitution rates. Accounting for gene and species specific rates allows SPIDIR to build gene trees more accurately than previous methods. A training set for learning rate parameters can be chosen from gene trees which are congruent to the species tree. An important algorithmic concern for gene tree reconstructions is devising a fast tree search method. In lecture, we saw how the tree search could be sped up by only computing the full $argmax L,T,RP(L,T,R|D,S,\Theta)$ for trees with high prior probabilites. This is accomplished through a computational pipeline where in each iteration 100s of trees are proposed by some heuristic. The topology prior $P(T,R|D,S,\Theta)$ can be computed quickly. This is used as a filter where only the topologies with high prior probabilities are selected as candidates for the full likelihood computation.

The performance of SPIDIR was tested on a real dataset of 21 fungi. SPIDER recovered over 96% of the synteny orthologs while other algorithms found less than 65%. As a result, SPIDER invoked much fewer number of duplications and losses.

26.6 Ancestral Recombination Graphs

TODO: Song, Yuns *@Next Year: Fill in this section based on: www.eecs.berkeley.edu/ yss/Pub/SH-JCB05.pdf and the course notes from 2012.*

26.6.1 The Sequentially Markov Coalescent

TODO: Song, Yuns *@Next Year: Fill this in based on: http://www.ncbi.nlm.nih.gov/pubmed/21270390*

26.7 Conclusion

Incorporating species tree information into the gene tree building process via introducing separate gene and species substitution rates allows for accurate parsimonious gene tree reconstructions. Previous gene tree reconstructions probably vastly overestimated the number of duplication and loss events. Reconstructing gene trees for large families remains a challenging problem.

26.8 Current Research Directions

26.9 Further Reading

26.10 Tools and Techniques

26.11 What Have We Learned?

Bibliography

[1] O. Akerborg, B. Sennblad, L. Arvestad, and J. Lagergren. Bayesian gene tree reconstruction and reconciliation analysis. *Proc Natl Acad Sci*, 106(14):5714–5719, Apr 2009.

[2] Zmasek C.M. and Eddy S.R. Analyzing proteomes by automated phylogenomics using resampled inference of orthologs. *BMC Bioinformatics*, 3(14), 2002.

[3] Li H, Coghlan A, Ruan J, Coin LJ, Heriche JK, Osmotherly L, Li R, Liu T, Zhang Z, Bolund L, Wong GK, Zheng W, DEhal P, Wang J, and Durbin R. Treefam: a curated database of phylogenetic trees of animal gene families. *Nucleic Acids Res*, 34, 2006.

[4] Arvestad L., Berglund A., Lagergren J., and Sennblad B. Bayesian gene/species tree reconciliation and orthology analysis using mcmc. *Bioinformatics*, 19 Suppl 1, 2003.

[5] M. D. Rasmussen and M. Kellis. A bayesian approach for fast and accurate gene tree reconstruction. *Mol Biol Evol*, 28(1):273290, Jan 2011.

[6] Matthew D. Rasmussen and Manolis Kellis. Accurate gene-tree reconstruction by learning gene and species-specific substitution rates across multiple complete genomes. *Genome Res*, 17(12):1932–1942, Dec 2007.

[7] C.E.V. Storm and E.L.L. Sonnhammer. Automated ortholog inference from phylogenetic trees and calculation of orthology reliability. *Bioinformatics*, 18(1):92–99, Jan 2002.

[8] Hollich V., Milchert L., Arvestad L., and Sonnhammer E. Assessment of protein distance measures and tree-building methods for phylogenetic tree reconstruction. *Mol Biol Evol*, 22:2257–2264, 2005.

[9] Wapinski, I. A. Pfeffer, N. Friedman, and A. Regev. Automatic genome-wide reconstruction of phylogenetic gene trees. *Bioinformatics*, 23(13):i549–i558, 2007.

CHAPTER

TWENTYSEVEN

POPULATION HISTORY

Guest Lecture by David Reich
Scribed by Brian Cass (2010)
Layla Barkal and Matt Edwards (2009)

Figures

27.1 Introduction

Humans share 99.9% of the same genetic information, and are 99% similar to chimpanzees. Learning about the 0.1% difference between humans can be used to understand population history, trace lineages, predict disease, and analyze natural selection trends. In this lecture, Dr. Reich explained how we can use this data to see evidence of gene flow between neanderthals and modern humans of Western Eurasian decent.

Last year, he examined India as a case of how genetic data can inform population history, which is included as an appendix.

27.2 Quick Survey of Human Genetic Variation

In the human genome, there is generally a polymorphism every 1000 bases, though there are regions of the genome where this rate can quadruple. These polymorphisms are markers of genetic variation. It is necessary to understand how genetic variation arises before attempting to analyze it. Single Nucleotide Polymorphisms (SNPs) are one manifestation of genetic variation. When SNPs occur, they are segregated according to recombination rates, advantages or disadvantages of the mutation, and the population structure that exists and continues during the lifespan of the SNP. Through the passing of generations, recombination splits the SNP haplotype into smaller blocks. The length of these blocks, then, is dependent on the rate of recombination and the stability of the recombination product. Therefore, the length of conserved haplotypes can be used to infer the age of mutation or its selection. An important consideration, though, is that the rate of recombination is not uniform across the genome; rather, there are recombination hot spots that can skew the measure of haplotype age or selectivity. This makes the haplotype blocks longer than expected under a uniform model.

To Dr. Reich, every place in the genome is best thought of as a tree when compared across individuals. But, depending on where are you look within the genome, this particular tree will be different than another particular tree you may get from a specific set of SNPs. The trick is to use the data that we have available on SNPs to infer the underlying trees, and then the overarching phylogenetic relationships. Take, for instance, the Y chromosome. It undergoes little to no recombination and thus can produce a high accuracy tree as it passed down from father to son. Likewise, we can take mitochondrial DNA, passed down from mother to child. While these trees can have high accuracy, other autosomal trees are confounded with recombination, and thus show lower accuracy to predict phylogenetic relationships. Gene trees are best made by looking at areas of low recombination, as recombination mixes trees. In general, there are about 1 to 2 recombinations per generation.

Humans show about 10,000 base-pairs of linkage, as we go back about 10,000 generations. Fruit fly linkage equilibrium blocks, on the other hand, are only a few hundred bases. Fixation will occur over time, proportional to the size of the population. For a population of about 10,000 it will take about 10,000 years to reach that point. When a population expands, genetic drift goes down. So, curiously enough, the variation in humans looks like what would have been formed in a population size of 10,000.

If long haplotypes are mapped to genetic trees, approximately half of the depth is on the first branch; most morphology changes are deep in the tree because there was more time to mutate. One simple model of mutation without natural selection is the Wright-Fisher neutral model which utilizes binomial sampling. In this model, a population will reach fixation (frequency 1), will die out (frequency 0), or continue to segregate. In the human genome, there are 10-20 million common SNPs. This is less diversity than chimpanzees, implying that humans are genetically closer to one another.

With this genetic similarity in mind, comparing human sub-populations can give information about common ancestors and suggest historical events. The similarity between two sub-populations can be measured by comparing allele frequencies in a scatter plot. If we plot the frequencies of SNPs across different populations on a scatterplot, we see more spread between more distant populations. The plot below, for example, shows the relative dissimilarity of European American and American Indian populations along with the greater similarity of European American and Chinese populations. The plots indicate that there was a divergence in the past between Chinese and Native Americans, evidence for the North American migration bottleneck that has been hypothesized by archaeologists. The spread among different populations within Africa is quite large. We can measure spread by Fst (which describes the variance).

Several current studies have shown that unsupervised clustering of genetic data can recover self-selected labels of ethnic identity.[3] In Rosenbergs experiment, a bayesian clustering algorithm was developed. They took a sample size of 1000 people (50 populations, 20 people per population), and clustered those people

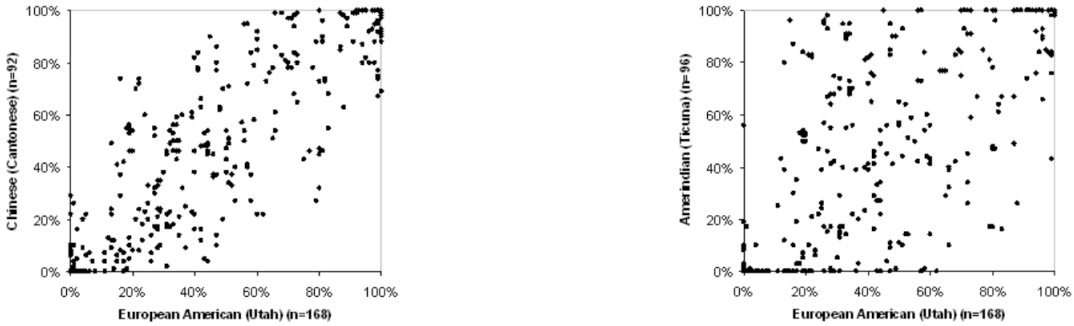


Figure 27.1: Similarity between two subpopulations can be measured by comparing allele frequencies in a scatterplot. The plots show the relative dissimilarity of European American and American Indian populations along with greater similarity of European American and Chinese populations.

by their SNP genetic data but they didnt tag any of the people by populations, so they could see how the algorithm would cluster them without knowledge of ethnicity. They didnt know what the optimal number of clusters was, so they tried 2, then 3, then 4, then 5, et cetera. What they found was that with 2 clusters, East-Asians and non-East-Asians were separated. With 3 clusters, Africans were separated from everyone else. With 4, East-Asians and Native Americans were separated. And then with 5, the smaller sub-populations began to emerge.

When waves of humans left Africa, genetic diversity decreased; the small numbers of people in the groups that left Africa allowed for serial founder events to occur. These serial founder events lead to the formation of sub-populations with less genetic diversity. This is evidenced by the fact that genetic diversity decreases moving out of Africa. West Africans have the highest diversity of any human sub-population.

27.3 Neanderthals and Modern Human Gene Flow

Recently, Dr. Reich worked with the Max Planck Institute as a population geneticist studying Neanderthal genetic data. He discussed with us the background of his research as part of the Neanderthal genome project, the draft sequence that they assembled, and then the evidence thats been compiled for gene flow between modern humans and Neanderthals.

27.3.1 Background

Clear fossils from Neanderthals from 200,000 years ago exist in West Eurasia (Europe and Western Asia), which is far earlier than *Homo erectus*. The earliest fossils of us come from Ethiopia dating about 200,000 years ago. However, there is evidence that neanderthals and humans overlapped in time and space between 135,000 and 35,000 years ago.

The first place of contact could have occurred in The Levant, in Israel. There are human fossils from 120,000 years ago, then a gap, neanderthal fossils about 80,000 years ago, another gap, and then human fossils again 60,000 years ago. This is proof of an overlap in place, but not in time. In the upper paleolithic era, there was an explosions of populations out of Africa (the migration about 60,000 to 45,000 years ago). In Europe after 45,000 years ago, there are sites where neanderthals and humans exist in the fossil record side by side. (Eastern Eurasia is not well documented, so we have little evidence from there.) Since there is

evidence that they co-existed, was there interbreeding? This is a question we wish to answer by examining population genomics.

Lets take a look at how you go about finding and sequencing DNA from ancient remains. First, you have to obtain a bone sample with DNA from a neanderthal. Human DNA and Neanderthal DNA is very similar (we are more similar to them than we are to chimps), so when sequencing short reads with very old DNA, its impossible to tell if the DNA is neanderthal or human. So, the cave is first classified as human or non-human, which helps to predict the origin of the bones. In sites of findings, typically lots of trash is left behind which is used to help classify the site (stone tools, particular technologies, the way meat was cut off animals, other trash). These caves are made up of lots of trash, and only the occasional bone. Even if you have a bone, it is still very unlikely that you have any salvageable DNA. In fact, 99% of the sequence of Neanderthals comes from only three long bones found in one site: the Vindija cave in Croatia (5.3 Gb, 1.3x full coverage). The paleontologists chose to sacrifice the long bones because they were less morphologically helpful.

Next, the DNA is sent to an ancient-DNA lab. Since they are 40,000 year old bones, there is very little DNA left in them. So, they are first screened for DNA. If they have it, is it primate DNA? Usually it is DNA from microbes and fungi that live in soil and digest us when we die. If it is primate DNA, is it contamination from the human (archeologist or lab tech) handling it? The difference between human and neanderthal DNA is 1/600 bp. The size of reads from a 40,000 year old bone sample is 30-40 bp. The reads are almost always identical for a human and neanderthal, so it is difficult to distinguish them.

Only about 1-10% of the DNA on old bones is the primates DNA. 89 DNA extracts were screened for neanderthals, but only 6 bones were actually sequenced (requires lack of contamination and high enough amount of DNA). The process of retrieving the DNA requires drilling beneath the bone surface (to minimize contamination) and taking samples from within. For the three long bones, less than 1 gram of bone powder was able to be obtained. Then the DNA is sequenced and aligned to a reference chimp genome. It is mapped to a chimp instead of a particular human because mapping to a human might cause bias if you are looking to see how the sequence relates to specific human sub-populations.

Most successful finds have been in cool limestone caves, where it is dry and cold and perhaps a bit basic. The best chance of preservation occurs in permafrost areas. Very little DNA is recoverable from the tropics. The tropics have a great fossil record, but DNA is much harder to obtain. Since most bones dont yield enough or good DNA, scientists have the screen samples over and over again until they eventually find a good one.

27.3.2 Draft Sequence

The neanderthal DNA had short reads, about 37 bp on average. There are lots of holes due to mutations caused by time eroding the DNA. However, there some characteristic mutations that occur on DNA thats been sitting for very long periods of time. There is a tendency to see C to T mutations, and G to A mutations. Over time, a methyl group gets knocked off of a C, which causes it to resemble to U. When PCR is used to amplify the DNA for sequencing, the polymerase sees a U and repairs it to a T. The G to A mutations are just the result of seeing that on the opposite strand, so really the important mutation to worry about is the C to T. This mutation is seen about 2% of the time! In order to combat this, scientists use a special enzyme now that recognizes the U, and instead of replacing it with a T, simply cuts the strand where it sees the mutation. This helps to identify those sites.

The average fragment size is quite small, and the error rate is still 0.1% - 0.3%. One way to combat the mutations is the note that on a double stranded fragment, the DNA is frayed towards the ends, where it becomes single stranded for about 10 bp. There tend to be high rates of mutations in the first and last 10

bases, but high quality elsewhere (so C to T mutations in the beginning and G to A in the end). In chimps, the most common mutations are transitions (purine to purine, pyrimidine to pyrimidine), and transversions are much rarer. The same goes for humans. Since the G to A and C to T mutations are transitions, it can be determined that there are about 4x more mutations in the Neanderthal DNA than if it were fresh by noting the number of transitions seen compared to the number of transversions seen (by comparing Neanderthal to human DNA). Transversions have a fairly stable rate of occurrence, so that ratio helps determine how much error has occurred through C to T mutations.

We are now able to get human contamination of artifact DNA down to around ¡1%. When the DNA is brought in, as soon as it is removed from the bone it is bar coded with a 4 bp tag (originally it was 4, now it is 7). That allows you to avoid contamination at any later point in the experiment, but not earlier. Extraction is also done in a clean room with UV light, after having washed the bone. Mitochondrial DNA is helpful for distinguishing what percent of the sample is contaminated with human DNA. Mitochondrial DNA is filled with characteristic event sites in different species - all Neanderthals were of one type, all humans of another (called reciprocally monophylogenetic). The contamination can be measured by counting the ratio of those sites. In the Neanderthal DNA, contamination was present, but it was ¡ 0.5%.

In sequencing, the error rate is almost always higher than the polymorphism rate. Therefore, most sites in the sequence that are different from humans are caused by sequencing errors. So we cant exactly learn about Neanderthal biology through the sequence generated, but we can analyze particular SNPs as long as we know where to look. The probability of a particular SNP being changed due to an error in sequencing is only $\frac{1}{300}$ to 11000. So we can, in fact, still get lots of usable data from this after all.

After aligning the chimp, Neanderthal, and modern human sequences, we can measure the distance that Neanderthals are, on the scale from humans to chimps. This distance is only about 12.7% from the human reference sequence. a French sample measures about 8% distance from the reference sequence, and a Bushman about 10.3%. What this says is that the Neanderthal DNA is within our range of variation as a species.

27.3.3 Evidence for Gene Flow

1. First, lets look at a comparison test. Take two randomly chosen populations, sequence both, and for each different SNP, check to see which population the Neanderthal DNA matched. This was done for 8 sequences. When Eurasians were compared with Eurasians, there was little difference. When Africans were compared with Africans, there was also little difference. However, when Africans were compared with non-Africans, Neanderthal SNPs much more highly matched the non-African DNA. This is evidence that there was mating and gene flow between Neanderthals and Eurasian modern humans.
2. Second, well take a look at a long range haplotype study done at Berkeley. These researchers picked long range sections of the genome and compared them among randomly chosen humans from various populations. When you look to see where the deepest branch of the tree constructed from that haplotype is, it almost always comes from an African population. However, occasionally non-Africans have the deepest branch. The study found that there were 12 regions where non-Africans have the deepest branch. When this data was used to analyze the Neanderthal genome, it was found that $\frac{10}{12}$ of these regions in non-Africans matched Neanderthals more than the matched the human reference sequence (a compilation of sequences from various populations). This is evidence of that haplotype actually being of Neanderthal origin.
3. Lastly, there is a bigger divergence than expect among humans. The average split between a Neanderthal and a human is about 800,000 years. The typical divergence between two humans is about 500,000 years. When looking at african and non-african sequences, regions of low divergence emerged in non-african sequences when compared with Neanderthal material. The regions found were highly

enriched for Neanderthal material (94% Neanderthal), which would increase the average divergence between humans (as the standard Neanderthal - human divergence is about 800,000 years).

27.4 Discussion

There was an example of a 50,000 year old bone found in southern Siberia, where the mtDNA was sequenced, that appears to be an out-group to both Neanderthals and modern humans. It was a little finger bone of a child. It is twice as deep in the phylogenetic tree as either of them, and has 1.9x coverage. These pieces of the ancestral DNA puzzle help us piece together human history and before. They serve to help us understand where we came from.

The bottleneck caused by the migration from Africa is only one example of many that have occurred. Most scientists usually concentrate on the age and intensity of events and not necessarily the duration, but the duration is very important because long bottlenecks create a smaller range of diversity. One way to help tell the length of a bottleneck is to determine if any new variations arose during it, as that occurs during longer bottlenecks, and as they will help distinguish how long it lasted. That change in range of diversity is also what helped create the different human sub-populations that became geographically isolated. This is just another way that population genomics can be useful for helping to piece together information.

Today, Dr. Reich showed how genetic differences between species (specifically here within primates) can be used to help understand the phylogenetic tree from which we are all derived. We looked at the case study of comparisons with Neanderthal DNA, learned about how ancient DNA samples are obtained, how sequences are found and interpreted, and how that evidence shows high likelihood of interbreeding between modern humans (of eurasian descent) and Neanderthals. Those very small differences between one species and the next, and within species, are what allow us to deduce a great deal of this history through population genetics.

27.5 Current Research Directions

27.6 Further Reading

27.6.1 Fall 2009 Discussion Topic: Genomic Variation in 25 Groups in India

There is a general taxonomy for studying population relationships with genetic data. The first general type of study utilizes both phylogeny and migration data. It fits the phylogenies to Fst values, values of sub-population heterozygosity (pioneered by Cavalli-Sforza and Edwards in 2267).**TODO: cite the paper where this is discussed in more detail** *@scribe: This* method also makes use of synthetic maps and Principal Components Analysis. [2] The primary downside to analyzing population data this way is uncertainty about results. There are mathematical and edge effects in the data processing that cannot be predicted. Also, certain groups have shown that separate, bounded mixing populations can produce significant-seeming principal components by chance. Even if the results of the study are correct, then, they are also uncertain.

The second method of analyzing sub-population relationships is genetic clustering. Clusters can be formed using self-defined ancestry [1] or the STRUCTURE database. [3] This method is overused and can over-fit the data; the composition of the database can bias the clustering results.

Technological advances and increased data collection, though, have produced data sets that are 10,000 times larger than before, meaning that most specific claims can be disproved by some subset of data. So in effect, many models that are predicted either by phylogeny and migration or genetic clustering will be disproved at some point, leading to large-scale confusion of results. One solution to this problem is to use a simple model that makes a statement that is both useful and has less probability of being falsified.

Past surveys in India have studied such aspects as anthropometric variation, mtDNA, and the Y chromosome. The anthropometric study looked at significant differences in physical characteristics between groups separated by geography and ethnicity. The results showed variation much higher than that of Europe. The mtDNA study was a survey of maternal lineage and the results suggested that there was a single Indian tree such that age of lineage could be inferred by the number of mutations. The data also showed that Indian populations were separated from non-Indian populations at least 40,000 years ago. Finally, the Y chromosome study looked at paternal lineage and showed a more recent similarity to Middle Eastern men and dependencies on geography and caste. This data conflicts with the mtDNA results. One possible explanation is that there was a more recent male migration. Either way, the genetic studies donein India have served to show its genetic complexity. The high genetic variation, dissimilarity with other samples, and difficulty of obtaining more samples lead to India being left out of HapMap, the 1000 Genomes Project, and the HGDP.

For David Reich and collaborators study of India, 25 Indian groups were chosen to represent various geographies, language roots, and ethnicities. The raw data included five samples for each of the twenty five groups. Even though this number seems small, the number of SNPs from each sample has a lot of information. Approximately five hundred thousand markers were genotyped per individual. Looking at the data to emerge from the study, if Principal Components Analysis is used on data from West Eurasians and Asians, and if the Indian populations are compared using the same components, the India Cline emerges. This shows a gradient of similarity that might indicate a staggered divergence of Indian populations and European populations.

27.6.2 Almost All Mainland Indian Groups are Mixed

Further analysis of the India Cline phenomenon produces interesting results. For instance, some Pakistani sub-populations have ancestry that also falls along the Indian Cline. Populations can be projected onto the principal components of other populations: South Asians projected onto Chinese and European principal components produces a linear effect (the India Cline), while Europeans projected onto South Asian and Chinese principal components does not. One interpretation is that Indian ancestry shows more variability than the other groups. A similar variability assessment appears when comparing African to non-African populations. Two tree hypotheses emerge from this analysis:

1. there were serial founder events in Indias history or

2. there was gene flow between ancestral populations.

The authors developed a formal four population test to test ancestry hypotheses in the presence of admixture or other confounding effects. The test takes a proposed tree topology and sums over all SNPs of (Pp1 Pp2)(Pp3 Pp4), where P values are frequencies for the four populations. If the proposed tree is correct, the correlation will be 0 and the populations in question form a clade. This method is resistant to several problems that limit other models. A complete model can be built to fit history. The topology information from the admixture graphs can be augmented with Fst values through a fitting procedure. This method makes no assumptions about population split times, expansion and contractions, and duration of gene flow, resulting in a more robust estimation procedure.

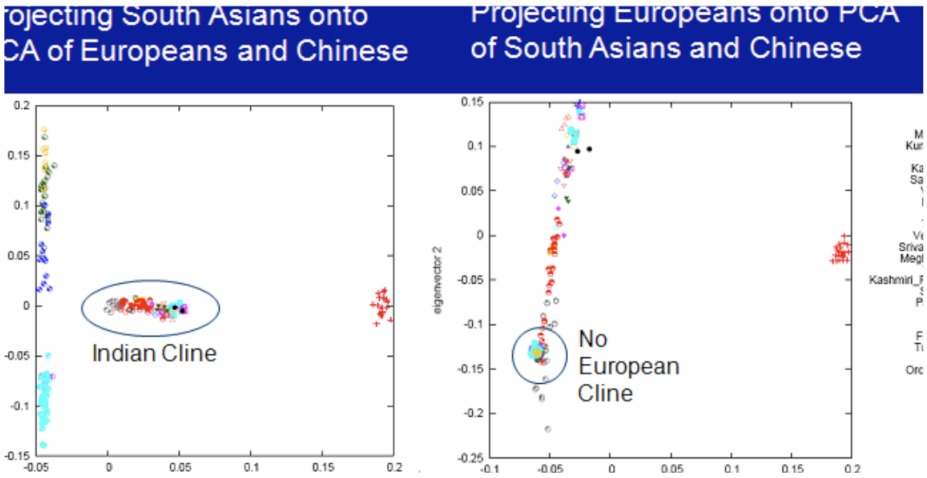


Figure 27.2: Populations can be projected onto the principal components of other populations: South Asians projected onto Chinese and European principal components produces a linear effect (the India Cline), while Europeans projected onto South Asian and Chinese principal components does not.

Furthermore, estimating the mixture proportions using the 4 population statistic gives error estimates for each of the groups on the tree. Complicated history does not factor into this calculation, as long as the topology as determined by the 4-population test is valid.

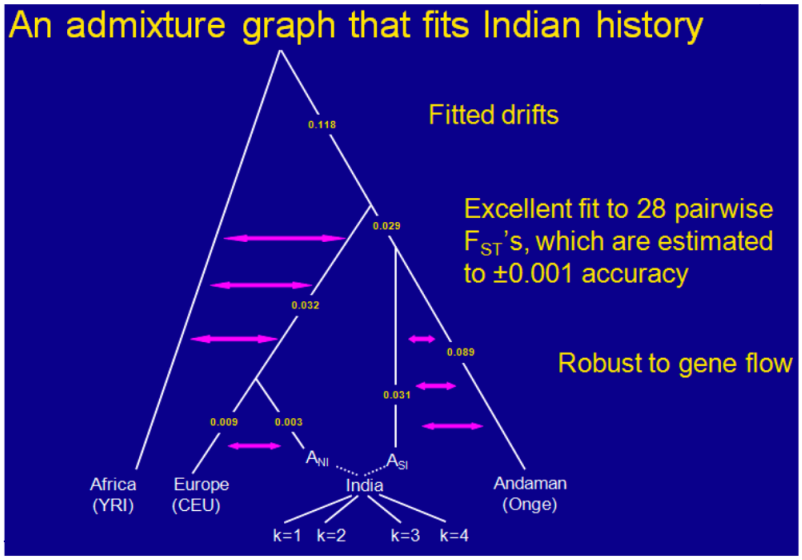


Figure 27.3: An admixture graph that fits Indian history

These tests and the cline analysis allowed the authors to determine the relative strength of Ancestral North Indian and Ancestral South Indian ancestry in each representative population sample. They found that high Ancestral North Indian ancestry is correlated with traditionally higher caste and certain language groupings. Furthermore, Ancestral North Indian (ANI) and South Indian (ASI) ancestry is as different from Chinese as European.

27.6.3 Population structure in India is different from Europe

Population structure in India is much less correlated with geography than in Europe. Even correcting populations for language, geographic, and social status differences, the Fst value is 0.007, about 7 times that of the most divergent populations in Europe. An open question is whether this could be due to missing (largely India-specific) SNPs on the genotyping arrays. This is because the set of targeted SNPs were identified primarily from the HapMap project, which did not include Indian sources.

Most Indian genetic variation does not arise from events outside India. Additionally, consanguineous marriages cannot explain the signal. Many serial founder events, perhaps tied to the castes or precursor groups, could contribute. Analyzing a single group at a time, it becomes apparent that castes and subcastes have a lot of endogamy. The autocorrelation of allele sharing between pairs of samples within a group is used to determine whether a founder event occurred and its relative age. There are segments of DNA from a founder, many indicating events more than 1000 years old. In most groups there is evidence for a strong, ancient founder event and subsequent endogamy. This stands in contrast to the population structure in most of Europe or Africa, where more population mixing occurs (less endogamy).

These serial founder events and their resulting structure have important medical implications. The strong founder events followed by endogamy and some mixing have lead to groups that have strong propensities for various recessive diseases. This structure means that Indian groups have a collection of prevalent diseases, similar to those already known in other groups, such as Ashkenazi Jews or Finns. Unique variation within India means that linkages to disease alleles prevalent in India might not be discoverable using only non-Indian data sources. A small number of samples are needed from each group, and more groups, to better map these recessive diseases. These maps can then be used to better predict disease patterns in India.

27.6.4 Discussion

Overall, strong founder events followed by endogamy have given India more substructure than Europe. All surveyed tribal and caste groups show a strong mixing of ANI and ASI ancestry, varying between 35% and 75% ANI identity. Estimating the time and mechanism of the ANI-ASI mixture is currently a high priority. Additionally, future studies will determine whether and how new techniques like the 4-population test and admixture graphs can be applied to other populations.

27.7 Tools and Techniques

27.8 What Have We Learned?

Bibliography

[1] Bowcock AM, Ruiz-Linares A, Tomfohrde J, Minch E, Kidd JR, and Cavalli-Sforza LL. High resolution of human evolutionary history trees with polymorphic microsatellites. *Nature*, 368:455–457, 1994.

[2] Menozzi. Synthetic maps of human gene frequencies in europeans. *Science*, 201(4358):768–792, Sep 1978.

[3] Rosenberg N. Genetic structure of human populations. *Science*, 298(5602):2381–2385, 2002.

CHAPTER

TWENTYEIGHT

POPULATION GENETIC VARIATION

Guest Lecture by Pardis Sabeti
Scribed by Mohammad Ghassemi, Jonas Helfer, Ben Mayne (2012),
Alex McCAuley (2010), Matthew Lee (2009), Arjun K. Manrai and Clara Chan (2008)

Figures

28.1 Introduction

For centuries, biologists had to rely on morphological and phenotypical properties of organisms in order to infer the tree of life and make educated guesses about the evolutionary history of species. Only recently, the ability to cheaply sequence entire genomes and find patterns in them has transformed evolutionary biology. Sequencing and comparing genomes on a molecular level has become a fundamental tool that allows us to gain insight into much older evolutionary history than before, but also to understand evolution at a much smaller resolution of time. With these new tools, we can not only learn the relationship between distant clades that separated billions of years ago, but also understand the present and recent past of species and even different populations inside a species.

In this chapter we will discuss the study of Human genetic history and recent selection. The methodological framework of this section builds largely on the concepts from previous chapters. Most specifically, the methods for association mapping of disease and phylogenetic constructs such as tree building among species and genes, and the history of mutations using coalescence. Having learned about these methods in the last chapter, we now will study how their application can inform us about the relationships, and differences between human populations. Additionally, we will look for how these differences can be exploited to look for signals of recent natural selection and the identification of disease loci. We will also discuss in this chapter what we currently know about the differences between human populations and describe some parameters we can infer that quantify population differences, using only the extent genetic variation we observe. In the study of Human Genetic history and recent selection, there are two principal topics of investigation which are often studied. The first is the history of population sizes. The second is the history of interactions between populations. Questions are often asked about these areas because the answers can often provide knowledge to improve the disease mapping process. Thus far, all present research based knowledge of human history was found by investigating functionally neutral regions of the genome, and assuming genetic drift. The reason that neural regions are employed is because mutations are subject to positive, negative and balancing selection pressure, when they take place on a functional region. Hence investigating a neural regions provides a selection unbiased proxy for the drift between species. In this chapter we will delve into some of the characteristics of selection process in humans and look for patterns of human variation in terms of cross species comparisons, comparison synonymous and non-synonymous mutations, and haplotype structure.

28.2 Population Selection Basics

28.2.1 Polymorphisms

Polymorphisms are differences in appearance amongst members of the same species. Many of them arise from mutations in the genome. These mutations, or genetic polymorphisms, can be characterized into different types.

Single Nucleotide Polymorphisms (SNPs)

- The mutation of only a single nucleotide base within a sequence. In most cases, these changes are without consequence. However, there are some cases where the mutation of a single nucleotide has a major effect.

- For example, is caused by a from A to T, that causes a change from glutamic acid (GAG) to valine (GTG) in hemoglobin.

Variable Number Tandem Repeats

- When a short sequence is repeated multiple times, DNA Polymerase can sometimes "slip", causing it to make either too many or too few copies of the repeat. This is called a .
- For example, **Huntingtons disease** that is caused by too many repeats of the trinucleotide CAG repeat in the HTT gene. Having more than 36 repeats can lead to gradual muscle control loss and severe neurological degradation. Generally, the more repeats there are, the stronger the symptoms.

Insertion/Deletion

- Through faulty copying or DNA-repair, or of one or multiple nucleotides can occur.
- If the insertion or deletion is inside an exon (the protein-coding region of a gene) and does not consist of a multiple of three nucleotides, a will occur.
- Prime example is deletions in the CFTR gene, which codes for chloride channels in the lungs and may cause **Cystic Fibrosis** where the patient cannot clear mucous in the lungs and causes infection

Did You Know?
DNA profiling is based on short variable number tandem repeats (STR). DNA is cut with certain restriction enzymes, resulting in fragments of variable length that can be used to identify an individual. Different countries use different (but often overlapping) loci for these profiles. In North America, a system based on 13 loci is used.

28.2.2 Allele and Genotype Frequencies

In order to understand the evolution of a species through analysis of alleles or genotypes, we must have a model of how the alleles are passed on from one generation to another. It is of immense importance that the reader has a firm intuition for the Hardy-Weinberg Principle and Wright fisher model before continuing. Hence, we will provide here a short reminder of modelling the history of mutations via the these methods. First introduced over a hundred years ago, the Wright-Fisher Model is a mathematical model of genetic drift in a population. Specifically, it describes the probability of obtaining k copies of a new allele p within a population of size N, with a non-mutant frequency of q, and what its expected frequency will be in successive generations.

Hardy-Weinberg Principle

The states that allele and genotype frequencies within a population will remain constant unless there is an outside influence that pushes them away from that equilibrium.

The Hardy-Weinberg principle is based on the following assumptions:

- The population observed is very large

- The population is isolated, i.e. there is no introduction of another subpopulation into the general population
- All individuals have equal probability of producing offspring
- All mating in the population is at random
- No random mutations occur in the population from one generation to the next
- Allele frequency drives future genotype frequency (Prevalent allele drives Prevalent genotype)

In a Hardy-Weinberg Equilibrium, for two alleles A and a, occurring with probability p and $q = 1p$, respectively, the probabilities of a randomly chosen individual having the homozygous AA or aa (pp or qq, respectively) or heterozygous Aa or aA (2pq) genotypes can be described by the equation:

$$p^2|2pq|q^2 = 1$$

This equation gives a table of probabilities for each genotype, which can be compared with the observed genotype frequencies using statistical error tests such as the chi-squared test to determine if the Hardy-Weinberg model is applicable. Figure 28.1 shows the distribution of genotype frequencies at different allele frequencies.

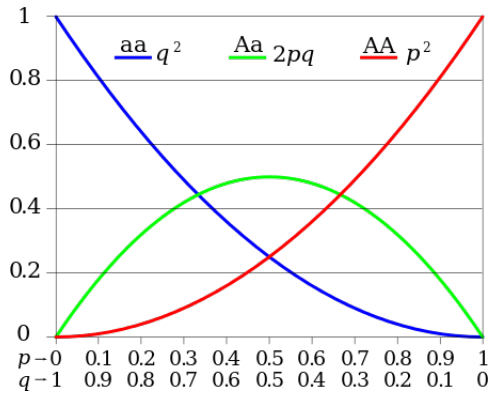


Figure 28.1: Plot of genotype frequencies for different allele frequencies

In natural populations, the assumptions made by the Hardy-Weinberg principle will rarely hold. Natural selection occurs, small populations undergo genetic drift, populations are split or merged, etc. In Nature a mutation will always either disappear (frequency = 0) from the population or become prevalent in a species - this is called fixation; in general, 99% of mutations disappear. Figure 28.2 shows a simulation of a mutations prevalence in a finite-sized population over time: both perform random walks, with one mutation disappearing and the other becoming prevalent:

Once a mutation has disappeared, the only way for it to reappear is the introduction of a new mutation into the population. For humans, it is believed that a given mutation under no selective pressure should fixate to 0 or 1 (within, e.g., 5%) within a few million years. However, under selection this will happen much faster.

Wright-Fisher Model

Under this model the time to fixation is 4N and the probability of fixation is 1/2N. In general Wright-Fisher is used to answer questions related to fixation in one way or another. To make sure your intuitions about

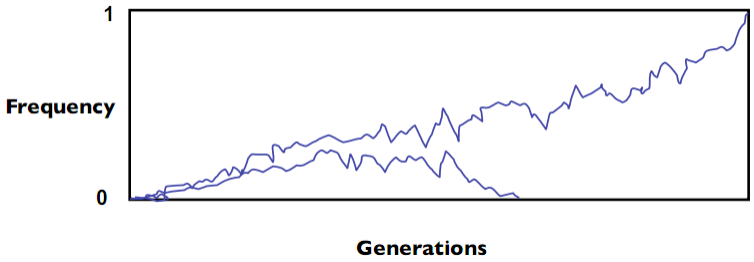


Figure 28.2: Changes in allele frequency over time

the method are absolutely clear considering the following questions:

FAQ

Q: Say you have a total of 5 mutations on a chromosome among a population of size 30, on average, how many mutations will be present in the next generation if each entity produces only one child?

A: If each parent has only one offspring, then there will be, on average, 5 mutations in the next generation because the expectation of allele frequencies is to remain constant according to the Hardy-Weinberg equilibrium principle in basic biology.

FAQ

Q: Is the Hardy-Weinberg Equilibrium principle's assumption about constant allele frequency reasonable?

A: No, the reality is far more complex as there is stochasticity in population size and selection at each generation. A more appropriate way to envision this is to image drawing alleles from a set of parents, with the amount of alleles in the next generation varying with the size of the population. Hence the frequency in the next generation could very well go up or down. Note here that if the allele frequency goes to zero it will always be at zero. The probability at each successive generation is lower if it's under negative selection and higher if it's under positive selection. Hence if it's a beneficial mutation the fixation time will be smaller, if the mutation is deleterious the fixation will be larger. If there are no offspring with a given mutation, then there won't be any decedents with that mutation either. If one produces multiple offspring however, who in turn produce multiple offspring of their own, then there is a greater chance that this allele frequency will rise.

FAQ

Q: Consider that the average human individual carries roughly 100 entirely unique mutations. So, when an individual produces offspring we could expect that half (or 50) of those mutations may appear in the child because in each sperm or egg cell, 50 of those mutations will be present, on average. Hence the offspring of an individual are likely to inherit approximately 100 mutations, 50 from one parent, and 50 from another in addition to their own unique mutations which come from neither parent. With this in mind, one might be interested in understanding what the chances are of some mutations appearing in the next generation if an individual produces, say, n children. How can one do this?

A: Hint: To compute this value, we assume that some allele originates in the founder, at some arbitrary chromosome (1 for example). Then we ask the question, how many chromosome 1s exist in the entire population? At the moment, the size of the human population is 7 Billion, each carrying two copies of chromosome 1.

The above questions and answers should make it painfully clear that the standard Hardy-Weinberg assumption of allele frequencies remaining constant from one generation to the next is violated in many natural cases including migration, genetic mutation, and selection. In the case of selection, this issue is addressed by modifying the formal definition to include a S, term which measures the skew in genotypes due to selection. See table 28.1 for a comparison of the original and selection compensated versions:

Behavior	With only drift	With drift and selection
n in next generation	Mean: $n(=2Np)$, Dist: $Binomial(2N,p)$	Mean: $n(1+\frac{s}{1+ps})$, Dist: $Binomial(2N, p\frac{1+s}{1+ps})$
Time to fixation	$4N$	$\frac{4N}{1+\frac{3}{8}N\|s\|}(\frac{1+\frac{1}{2}(\ln N)\|s\|}{1+\|s\|})$
Probability of fixation	$\frac{1}{2N}$	$\frac{1-e^{-2s}}{1-e^{-4Ns}}$

Table 28.1: Comparison of Wright-Fisher Model With Drift, Versus Drift and Selection

The main point to take away from Table 28.1, and this section of the chapter is that weather you have selection or not, it is highly unlikely that a single allele will fixate in a population. If you have a very small population, however, then the chances of an allele fixating are much better. This is often the case in human populations, where there are often small, interbred populations which allow for mutations to fix in a population after only a few generations, even if the mutation is deleterious in nature. This is precisely why we tend to see recessive deleterious mandolin disorders in isolated populations.

28.2.3 Ancestral State of Polymorphisms

How can we determine for a given polymorphism which version was the and which one is the mutant? The ancestral state can be inferred by comparing the genome to that of a closely related species (e.g. humans and chimpanzees) with a known phylogenetic tree. Mutations can occur anywhere along the phylogenetic tree sometimes mutations at the split fix differently in different populations ("fixed difference"), in which case the entire populations differ in genotype. However, recent mutations will not have had enough time to become fixed, and a polymorphism will be present in one species but fully absent in the other as simultaneous mutations in both species are very rare. In this case, the "derived variant" is the version of the polymorphism appearing after the split, while the ancestral variant is the version occuring in both species.

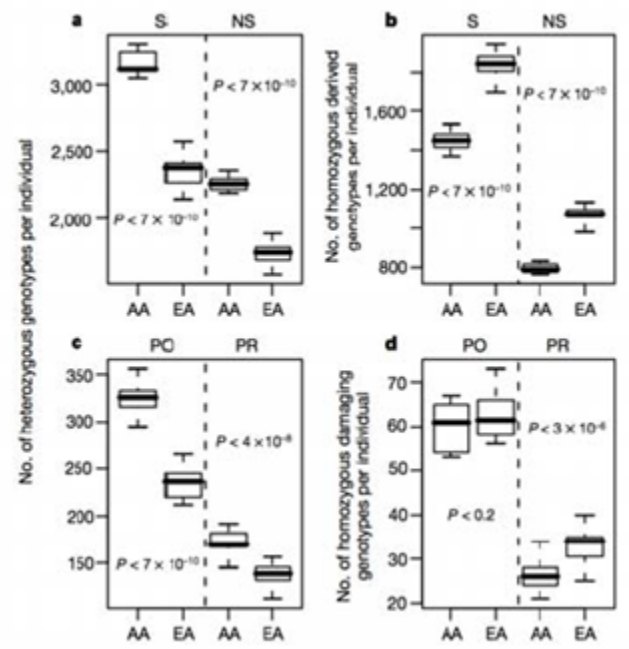


Figure 28.3: A comparison of the hetrozygous and homozygous derived and damaging genotypes per individual in an African American (AA) and European American (EA) population study.

28.2.4 Measuring Derived Allele Frequencies

The the frequency of the derived allele in the population can be easily calculated, if we assume that the population is homogeneous. However, this assumption may not hold when there is an unseen divide between two groups that causes them to evolve separately as shown in figure 28.4.

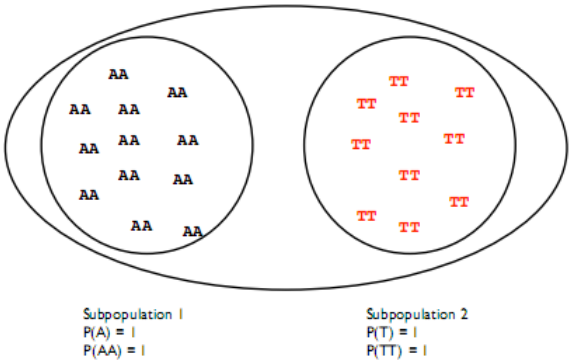


Figure 28.4: Two isolated populations

In this case the prevalence of the variants among subpopulations is different and the Hardy-Weinberg principle is violated.

One way to quantify this difference is to use the (F_{st}) to compare subpopulations within a species. In reality only a portion of the total heterozygosity in a species is found in a given subpopulation. Fst estimates the reduction in heterozygosity (2pq with alleles p and q) expected when 2 different populations are erroneously grouped together. Given a population having n alleles with frequencies p_i where ($1 \leq i \leq n$),

the homozygosity G of the population is calculated as:

$$\Sigma^{n}{}_{i=1} {p_i}^2$$

The total heterozygosity in the population is given by 1-G.

$$F_{st} = \frac{Heterozygosity(total) - Heterozygosity(subpopulation)}{Heterozygosity(total)}$$

In the case shown in figure 28.4 there is no heterozygosity between the populations, so $F_{st} = 1$. In reality the Fst will be small within one species. In humans, for example, it is only 0.0625. For in practise, the Fst is computed either by clustering sub-populations randomly or using an obvious characteristic such as ethnicity or origin.

28.3 Genetic Linkage

In the simple models we've seen so far, alleles are assumed to be passed on independently of each other. While this assumption generally holds in the long term, in the short term we will generally observe a that certain alleles are passed on together more frequently than expected. This is termed genetic linkage.

The , also known as Mendel's second law states:

Alleles of different genes are passed on independently from parent to offspring.

When this "law" holds, there is no correlation between different polymorphisms and the probability of a haplotype (a given set of polymorphisms) is simply the product of the probabilities of each individual polymorphism.

In the case where the two genes lie on different chromosomes this assumption of independence generally holds, but if the two genes lie on the same chromosome, they are more often than not passed on together. Without genetic recombination events, in which segments of DNA on homologous chromosomes are swapped (crossing-over), the alleles of the two genes would remain perfectly correlated. With however, the correlation between the genes will be reduced over several generations. Over a suitably long time interval, recombination will completely remove the linkage between two polymorphisms; at which point they are said to be in equilibrium. When, on the other hand, the polymorphisms are correlated, we have **Linkage Disequilibrium** (LD). The amount of disequilibrium is the difference between the observed haplotype frequencies and those predicted in equilibrium.

The linkage disequilibrium can be used to measure the difference between observed and expected assortments. If there are two alleles (1 and 2) and two loci (A and B) we can calculate haplotype probabilities and find the expected allele frequencies.

- Haplotype frequencies
  - $P(A_1) = x_{11}$

  - $P(B_1) = x_{12}$
  - $P(A_2) = x_{21}$
  - $P(B_2) = x_{22}$
- Allele frequencies
  - $P_{11} = x_{11} + x_{12}$
  - $P_{21} = x_{21} + x_{22}$
  - $P_{12} = x_{11} + x_{21}$
  - $P_{22} = x_{12} + x_{22}$
- $D = P_{11} * P_{22} P_{12} * P_{21}$

D_{max}, the maximum value of D with given allele frequencies, is related to D in the following equation:

$D' = \frac{D}{D_{max}}$

D' is the maximum linkage disequilibrium or complete skew for the given alleles and allele frequencies. D_{max} can be found by taking the smaller of the expected haplotype frequencies $P(A_1, B_2)$ or $P(A_2, B_1)$. If the two loci are in complete equilibrium, then $D' = 0$. If $D' = 1$, there is full linkage.

The key point is that relatively recent mutations have not had time to be broken down by crossing-overs. Normally, such a mutation will not be very common. However, if it is under positive selection, the mutation will be much more prevalent in the population than expected. Therefore, by carefully combining a measure of LD and derived allele frequency, we can determine if a region is under positive selection.

Decay of is driven by recombination rate and time (in generations) and has an exponential decay. For a higher recombination rate, linkage disequilibrium will decay faster in a shorter amount of time. However, the background recombination rate is difficult to estimate and varies depending on the location in the genome. Comparison of genomic data across multiple species can help in determining these background rates.

28.3.1 Correlation Coefficient r^2

Answers how predictive an allele at locus A is of an allele at locus B

$$r^2 = \frac{D^2}{P(A_1)P(A_2)P(B_1)P(B_2)}$$

As the value of r^2 approaches 1, the more two alleles at two loci are correlated. There may be linkage disequilibrium between two haplotypes, even if the haplotypes are not correlated at all. The correlation coefficient is particularly interesting when studying associations of diseases with genes, where knowing the genotype at locus A may not predict a disease whereas locus B does. There is also the possibility where neither locus A nor locus B are predictive of the disease alone but loci A and B together are predictive.

28.4 Natural Selection

In the mid 1800s the concept of evolution was not an uncommon idea, but it wasn't before Darwin and Wallace proposed natural selection as the mechanism that drives evolution in nature that the theory of

evolution got widespread recognition. It took 70 years (1948) until J.B.S Haldanes Malaria Hypothesis found the first example for natural selection in humans. He showed a correlation between genetic mutations in red blood cells and the distribution of malaria prevalence and discovered that individuals who had a specific mutation that made them suffer from sickle cell anaemia also gave made them resistant to malaria.

Lactose tolerance (lasting into adulthood) is another example of natural selection. Such explicit examples were hard to prove without genome sequences. With whole genome sequencing readily available, we can now search the genome for regions with the same patterns as these known examples to identify further regions undergoing natural selection.

28.4.1 Genomics Signals of Natural Selection

- K_a/K_s ratio of non-synonymous to synonymous changes per gene
- Low diversity and many rare alleles over a region (ex Tajima's D with regard to sickel-cell anemia)
- High derived allele frequency (or low) over a region (ex Fay and Wu's H)
- Differentiation between populations faster than expected from drift (Measured with F_{st})
- Long haplotypes: evidence of selective sweep.
- Exponential prevalence of a feature in sequential generations
- Mutations that help a species prosper

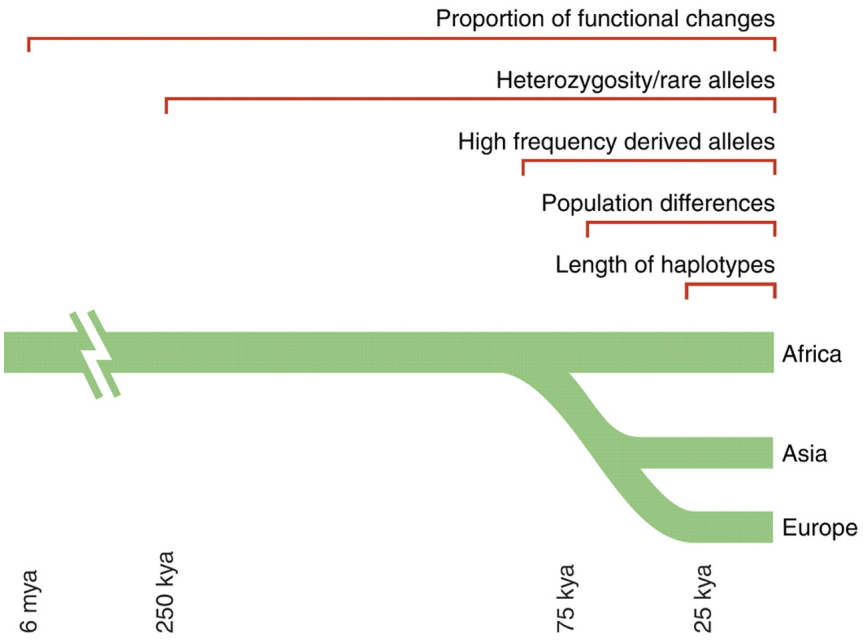


Figure 28.5: Approximate Time Table of Effects
Sabeti et al. *Science* 2006

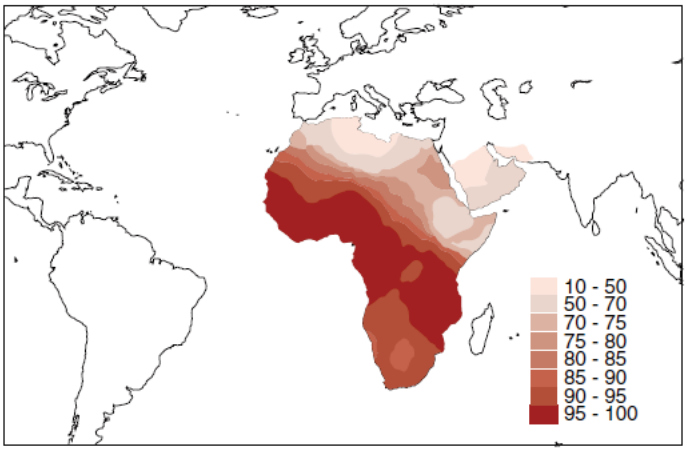
Examples of Negative (Purifying) Selection

- **Across species** we see negative selection of new mutations in conserved functional elements (exons, etc.).

- **New alleles** within one species tend to have lower allele frequencies if the allele is non-synonymous than synonymous. Lethal alleles have very low frequencies.

Examples of Positive (Adaptive) Selection

- **Similar to negative selection** in that positive selection more likely in functional elements or non-synonymous alleles.
- **Across species** in a conserved element, a positively selected mutation might be the same over most mammals, but change in a specific species because a positvely selected mutation appeared after speciation or caused speciation.
- **Within a species** positvely selected alleles likely differ in allele frequency (F_{st}) across populations. Examples include malaria resistance in African populations (28.6) and lactose persistence in European populations (28.7).
- **Polygenic selection** within species can arise when a trait is selected for that depends on many genes. An example is human height where 139 SNPs are known to be related to height. Most are not population specific mutations but alleles across all humans that are seleced for in some populations more than others. (28.8)



Extreme population differences in *FY*O* allele frequency. The *FY*O* allele, which confers resistance to *P. vivax* malaria, is prevalent and even fixed in many African populations, but virtually absent outside Africa (*38*).

Figure 28.6: Localized positive selection for Malaria resistance within species
Sabeti et al. *Science* 2006

Statistical Tests

- **Long range correlations (iHs, Xp, EHH):** If we tag genetic sequences in an individual based on their ancestry, we end up with a broken haplotype, where the number of breaks (color changes) is correlated with the number of recombinations and can tell us how long ago a particular ancestry was introduced.
- **SWEEP** A program developed by Pardis Sabeti, Ben Fry and Patrick Varilly. SWEEP detects evidence of natural selection by analyzing haplotype structures in the genome using the long range

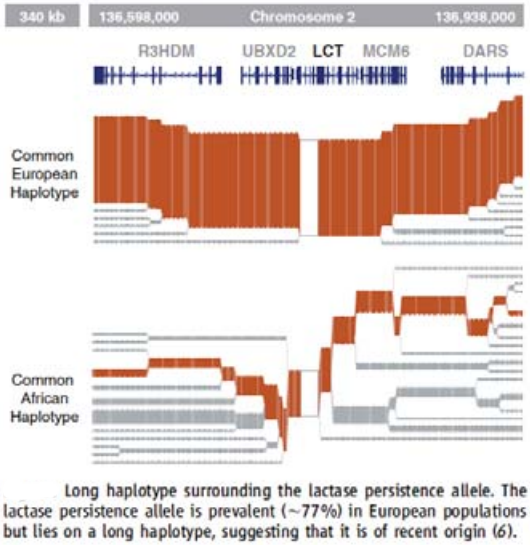


Figure 28.7: Localized positive selection for lactase persistence allele
Sabeti et al. *Science* 2006

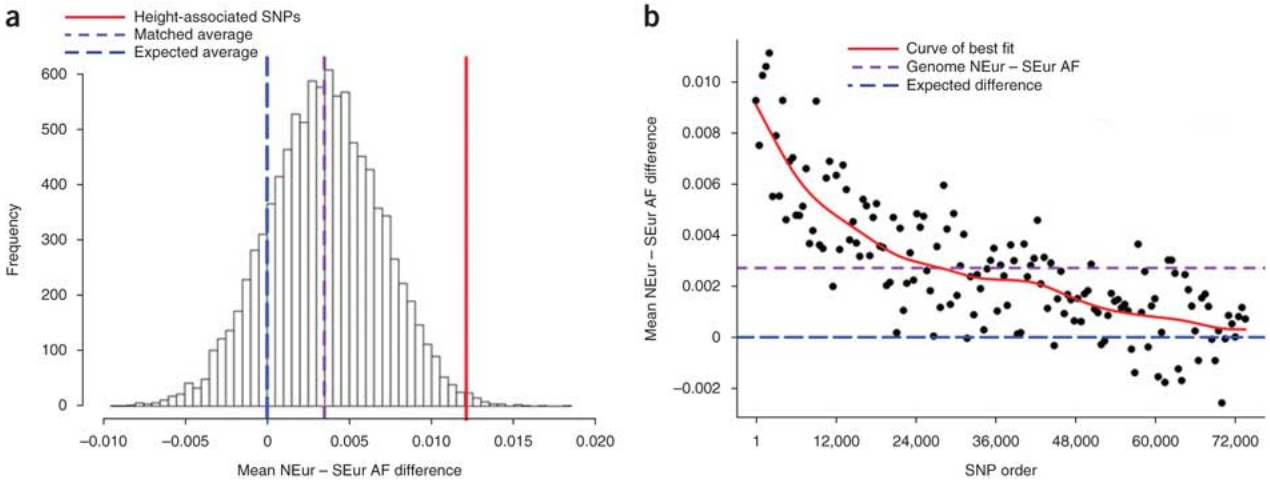


Figure 28.8: Mean allele frequency difference of height SNPs, matched SNPS, and genome-wide SNPS between Northern- and Southern-European populations
Turchin et al., Nature *Genetics* (2012)

haplotype test (LRH). It looks for high frequency alleles with long range linkage disequilibrium that hints to large scale proliferation of a haplotype that occurred at a rate greater than recombination could break it from its markers .

- **High Frequency Derived Alleles** Look for large spikes in the frequency of derived alleles in set positions.
- **High Differentiation (F_{st})** Large spikes in differentiation at certain positions.

Using these tests, we can find genomic regions under selective pressure. One problem is that a single SNP under positive selection will allow nearby SNPs to piggy-back and ride along. It is difficult to distinguish the SNP under selection from its neighbours with only one test. Under selection, all the tests are strongly

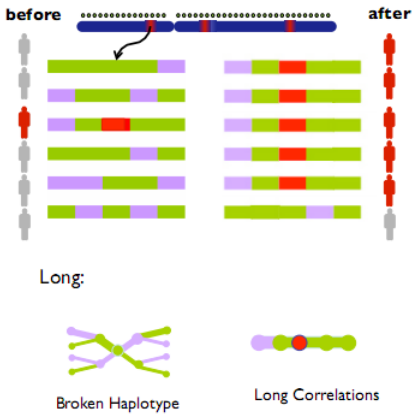


Figure 28.9: Broken haplotype as a signal of natural selection

correlated; however, in the absence of selection they are generally independent. Therefore, by employing a composite statistic built from all of these tests, it is possible to isolate the individual SNP under selection.

Examples where a single SNP has been implicated in a trait:

- Chr15 Skin pigmentation in Northern Europe
- Chr2 Hair traits in Asia
- Chr10 Unknown trait in Asia
- Chr12 Unknown Trait in Africa

28.5 Human Evolution

28.5.1 A History of the Study of Population Dynamics

Not surprisingly, the scientific community has a long, and somewhat controversial history of interest in recent population dynamics. While indeed some of this interest was applied toward more nefarious aims, such as the scientific justifications for racism for eugenics but these are increasingly the exception and not the rule. Early studies of population dynamic were primitive in many ways. Quantifying the differences between human populations was originally performed using blood types, as they seemed to be phenotypically neutral, could be tested for outside of the body, and seemed to be polymorphic in many different human populations. Fast forward to the present, and the scientific community has realized that there are other glycoproteins beyond the A,B and O blood groups that are far more polymorphic in the population. As science continued to advance and sequencing became a reality, they began whole genome sequencing of the Y-chromosome, mitochondrial and microsatellite markers around them. What's special about those two types of genetic data? First and foremost, they are quite short so they can be sequenced more easily than other chromosomes. Beyond just the size, the reason that the Y and mitochondrial chromosomes were of such interest is because they do not recombine, and can be used to easily reconstruct inheritance trees. This is precisely what makes these chromosomes special relative to a short chunk on an autosome; we know exactly where it comes from because we can trace paternal or maternal lineage backward in time.

This type of reconstruction does not work with other chromosomes. If one were to generate a tree using a certain chunk of all of chromosome 1 in a certain population, for instance, they would indeed form a phylogeny but that phylogeny would be picked from random ancestors in each of the family trees.

As sequencing continued to develop and grow more effective, the human genome project was being proposed, and along with it there was a strong push to include some sort of diversity measure in genomic data. Technically speaking, it was easiest to simply look at microsatellites for this diversity measure because they can be studied on gel to see size polymorphisms instead of inspecting a sequence polymorphism. As a reminder, a microsatellite is a region of variable length in the human genome often characterised by short tandem repeats. One reason for microsatellites is retroviruses inserting themselves into the genome, such as the ALU elements in the human genome. These elements sometimes become active and will retro-transpose as insertion events and one can trace when those insertion events have happened in human lineage. Hence, there was a push, early on to assay these parts of the genome in a variety of different populations. The really attractive thing about microsatellites is that they are highly polymorphic and one can actually infer their rate of mutation. Hence, we can not only say that there is a certain relationship between populations based on these rates, but we can also say how long they have been evolving and even when certain mutations occurred, and how long it's been on certain branches of the phylogenetic tree.

FAQ

Q: Can't this simply be done with SNPs

A: You can't do it very easily with SNPs.

You can get an idea of how old they are based on their allele frequency, but they're also going to be influenced by selection.

After the human genome project, came the Haplotype inheritance Hapmap project which looked at SNPs genome wide. We have discussed Haplotype inheritance in detail in prior chapters where we learned the importance of Hapmap in designing genotyping arrays which look at SNPs that mark common haplotypes in the population.

The effects of Bottlenecks on Human diversity Using this wealth of data across studies and a plethora of mathematical techniques has led to the realization that humans, in fact, have a very low diversity given our census population; which implies a small effective population size. Utilizing the Wright-Fisher model it is possible to work back from the level of diversity and the number of mutations we see in the population today to generate a founding population size. When this computation is performed it works out to being around 10,000.

FAQ

Q: Why is this so much smaller than our census population size?

A: There was A population bottleneck somewhere.

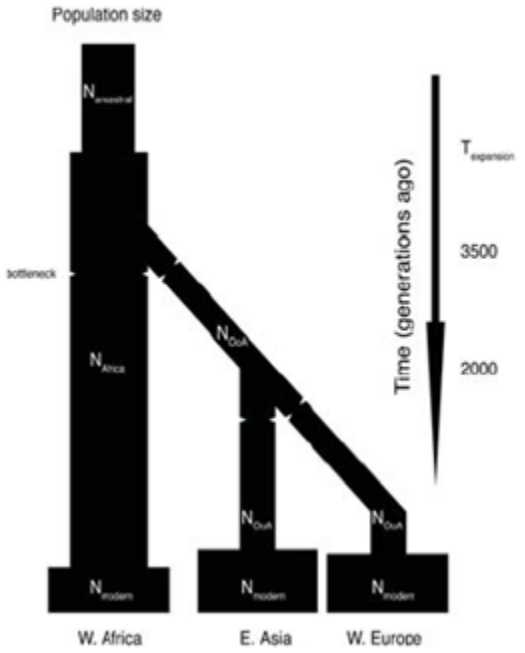


Figure 28.10: A depiction of two major bottleneck events, one in the founding population from Africa, and other, smaller subsequent bottleneck events in the East Asian and Western European populations.

Most of the total variation between humans is happening within-continent. One can measure how much diversity is explained by geography and how much is not. It turns out that most of it is not explained by geography. In fact, most common variants are polymorphic in every population and if a common variant is unique to a given population, there probably hasn't been enough time for that to happen by drift itself. Recall what an unlikely process it is to get to a high allele frequency over the course of several generations by mere chance alone. Hence, we may interpret this as a signal of selection when it occurs. All of the evidence in terms of comparing diversity patterns and trees back to ancestral haplotypes converges to an Out-of-Africa hypothesis which is the overwhelming consensus in the field and is the lens through which we review all the genetic population data. Starting from the African founder population, there have been works which have demonstrated that it's possible to model population growth using the wright fisher model. The studies have shown that the growth rate we see in Asian and European populations are only consistent with large exponential growth after the out-of-Africa event.

Study	Sample size (n)*	Time growth started (years ago)†	Initial N_e‡	Growth per generation (%)
Gravel *et al.* (5)	60	23,000§ (21,000–27,000)	1032 (677–1290)	0.48 (0.30–0.75)
Gutenkunst *et al.* (6) (including New World modeling)	22	26,400§ (21,700–30,700)	1500 (900–2200)	0.23 (0.16–0.34)
Gutenkunst *et al.* (6) (excluding New World modeling)	22	21,200§ (17,600–23,900)	1000 (500–1500)	0.4 (0.26–0.57)
Schaffner *et al.* (29)	62	8750‖	7700	0.73‖
Coventry *et al.* (18)	10,422	1400 (900–2800)	7700#	9.4 (4.5–14.5)

Table 28.2: Genetic Estimates of Recent Population Growth in Europe

This helps us understand the reasons for phonotypical differences between the races as Bottlenecks which

are followed by exponential growth can lead to an excess of rare alleles. The present theory on human diversity states that there were secondary bottleneck events after the founding population migrated out of Africa. These founders were, at some earlier point subject to an even smaller bottleneck event which is now reflected in every human genome on the planet, regardless of their immediate ancestry. It is possible to estimate how small the original bottle neck was by looking at differences between African and European origin individuals, inferring the effects of the secondary bottleneck, and the term of exponential growth of the European population. The other way of approaching bottleneck event estimation is to simply inspect the allele frequency spectrum needed to build coalescent trees. In this way, one can take haplotypes across the genome and ask what the most recent common ancestor was by observing how the coalescence varies across the genome. For instance, one may guess that some haplotype was positively selected for only recently given the length of the haplotype. An example of one such recent mutation in the European population is the lactase gene. Another example for the Asian population is the ER locus.

There is a wealth of literature showing that when one draws a coalescence tree for most haplotypes it ends up going way back before when we think speciation happened. This indicates that certain features have been kept polymorphic for a very long time. One can, however, look at this distribution of features across the whole genome and infer something about population history from it. If there was a recent bottle neck in a population, it will be reflected by the ancestors being very recent whereas more ancient things will have survived the bottleneck. One can take the distribution of coalescent times and run simulations for how the effect of population size would have varied with time. The model for doing this type of study was outlined by Li and Durbin. The Figure 28.11 from their study illustrates two such bottleneck events. The first is the bottleneck which occurred in Africa long before migrations out of the continent. This was then followed by a population specific bottleneck that resulted from migration groups out of Africa. This is reflected in the diversity of the populations today based on their ancestry and it can be derived from looking at a pair of chromosome from any two people in these populations.

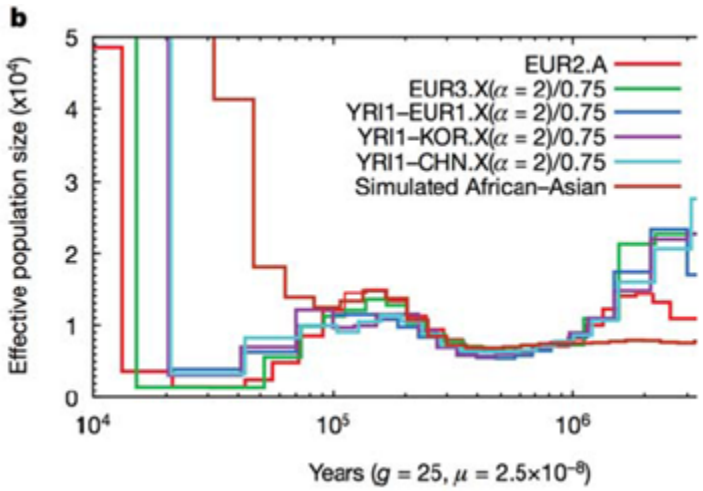


Figure 28.11: An illustration of two bottleneck events

28.5.2 Understanding Disease

Understanding that human populations went through bottlenecks has important implications for understanding population specific disease. A study published by Tennessen et al. this year was looking at exome sequences in many classes of individuals. The study intended to look at how rare variants might be contributing to disease and as a consequence they were able to fit population genetics models to the data, and ask what sort of deleterious variants were seen when sequencing exomes from a broad population panel. Using this approach, they were then able to generate parameters which describe how long ago exponential

growth between the founder, and branching populations occured. See figure 28.12 below for an illustration of this:

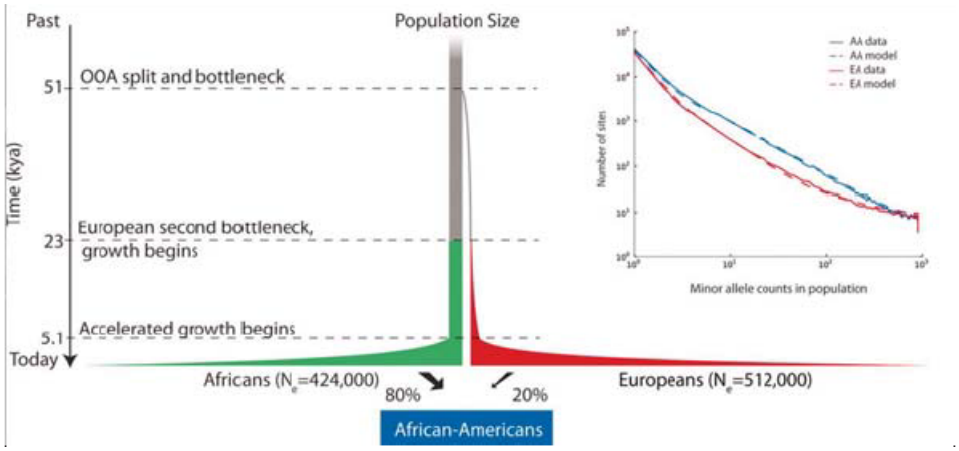


Figure 28.12: The figure illustrate the effects of a bottleneck events on the number of rare Alleles in a population.

28.5.3 Understanding Recent Population Admixture

In addition to viewing coalescent times, one can also perform Principal Component Analysis on SNPs to gain an understanding of more recent population admixtures. Running this on most populations shows clustering with respect to geographical location. There are some populations, however, that experienced a recent admixture for historical reason. The two most commonly referred to in the scientific literature are: African Americans, who on average are 20

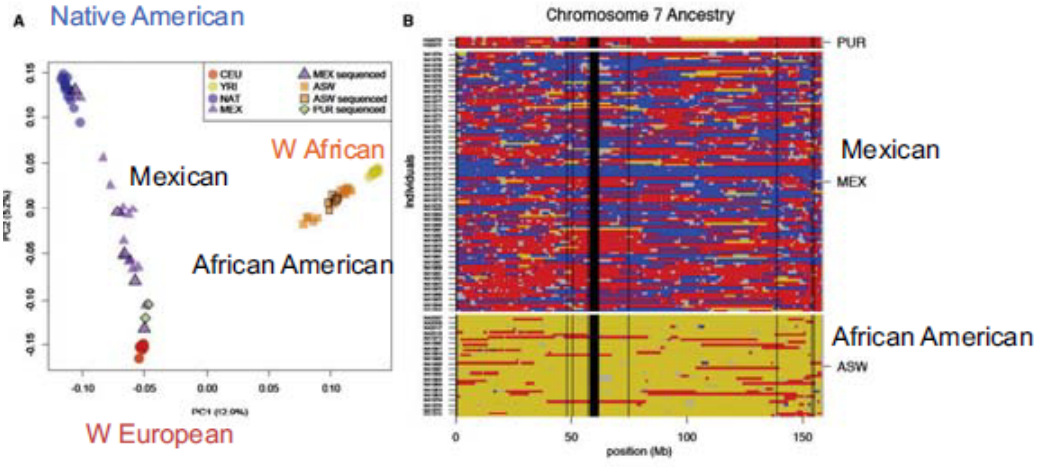


Figure 28.13: A depiction of European admixture levels in the Mexican, and African American populations.

There are two major things one can say about the admixture event of African Americans and Mexican Americans. The first and more obvious is inferring the admixture level. The second, and more interesting, is inferring when the admixture event happened based on the actual mixture level. As we have discussed in previous chapters, the racial signifiers of the genome break down with admixture because of recombination in each generation. If the population is contained, the percentage of those with European and West African

origin should stay the same in each generation, but the segments will get shorter, due to the mixing. Hence, the length of the haplotype blocks can be used to date back to when the mixing originally happened. (When it originally happened we would expect large chunks, with some gambits being entirely of African origin, for instance.) Using this approach, one can look at the distribution of recent ancestry traps and then fit a model to when these migrants entered an ancestral population as shown below:

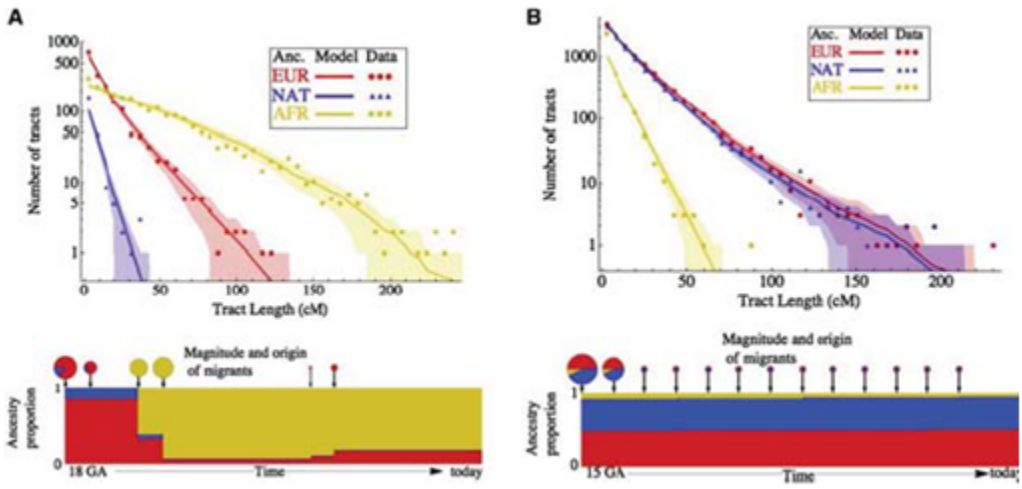


Figure 28.14: As illustration of the magnitude and origin of migrants based on the tract length and number of tracts in the admixed population.

28.6 Current Research

28.6.1 HapMap project

The International Project aims to catalog the genomes of humans from various countries and regions and find similarities and differences to help researchers find genes that will benefit the advance in disease treatment and administration of health related technologies.

28.6.2 1000 genomes project

The 1000 Genomes Project is an international consortium of researchers aiming to establish a detailed catalogue of human genetic variation. Its aim was to sequence the genomes of more than a thousand anonymous participants from a number of different ethnic groups. In October 2012, the sequencing of 1092 genomes was announced in a Nature paper. It is hoped that the data collected by this project will help scientists gain more insight into human evolution, natural selection and rare disease-causing variants.

28.7 Further Reading

- Campbell Biology, 9th edition; *Pearson*; Chapter 23: The Evolution of Populations

- The Cell, 5th edition, *Garland publishing*; Chapters 5: DNA replication, repair and recombination, Chapter 20: Germ cells and fertilization

Bibliography

CHAPTER

TWENTYNINE

MEDICAL GENETICS – THE PAST TO THE PRESENT

Guest Lecture by
Mark J. Daly (PhD)
Scribed by Anna Ayuso, Abhishek Sarkar (2011), Joel Brooks (2012)

Figures

29.1 Introduction

Mark J. Daly, Ph.D., is an Associate Professor at the Massachusetts General Hospital/Harvard Medical School and an Associate Member of the Broad Institute. This lecture explains how statistical and computational methods can aid researchers in understanding, diagnosing, and treating disease. The problem of identifying genetic variation which can explain phenotypic variation is known as association mapping. This problem is particularly important to solve for disease phenotypes (e.g., susceptibility). Historically, the method of choice for solving this problem was linkage analysis. However, advances in genotyping technology have allowed for a more powerful method called genome-wide association. More recent advances in genomic data have allowed for novel integrative analyses which can make more powerful predictions about diseases.

29.2 Goals of investigating the genetic basis of disease

Any discussion about the basis of disease must consider both genetic and environmental effects. However, it is known that many traits, for example those in Figure 29.1, have significant genetic components. Formally, the *heritability* of a phenotype is the proportion of variation in that phenotype which can be explained by genetic variation. The traits in Figure 29.1 are all at least 50% heritable.

Accurately estimating heritability involves statistical analyses on samples with highly varied levels of shared genetic variation (e.g., twins, siblings, relatives, and unrelated). Studies on the heritability of Type 2 diabetes, for example, have shown that given you have diabetes, the risk to the person sitting next to you (an unrelated person) increases by 5–10%; the risk to a sibling increases by 30%; and the risk to an identical twin increases by 85%–90%.

Having established that there is a genetic component to disease traits of interest, what are the goals of understanding this component? There are three main goals:

- Directing downstream research in disease
- Potential for improved diagnostics
- Enabling rational drug development

Identifying genetic variants which explain variation in the disease trait obviously contributes to our ability to understand the mechanism (the biochemical pathways, etc.) by which the disease manifests. Moreover, those variants can be used in genetic screens to test for increased risk for the disease trait. But the last goal is of particular interest because strong evidence suggests we do not really know how to develop effective drugs to target particular diseases. For example, in the last 50 years, no truly novel compounds have been developed to treat various psychiatric disorders such as schizophrenia.

Figure 29.2 depicts the cycle of drug development. First, researchers hypothesize a possible target of interest that might be related to a disease. They evaluate the biochemistry of this target, test the target in model organisms, and then finally perform clinical trials in humans. However, the vast majority of drugs which make it through this process end up being ineffective in treating the disease for which they were

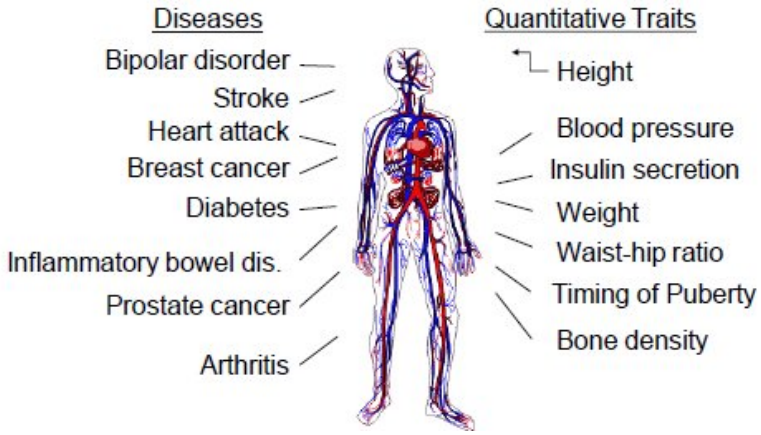


Figure 29.1: Examples of diseases and quantitative traits which have genetic components

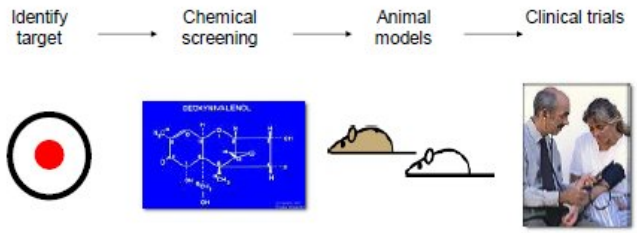


Figure 29.2: The drug development process

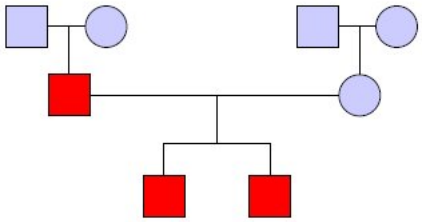


Figure 29.3: A pedigree which shows the inheritance of some trait

originally designed. This result is mainly a consequence of poor hypotheses about the basis of the disease in question.

Statins are a prominent example of highly effective drugs developed after work on understanding the genetic basis of the disease trait they are targeted at. Dr. Michael Brown and Dr. Joseph Goldstein won the Nobel Prize in Physiology or Medicine in 1985 for their work on the regulation of LDL cholesterol metabolism [5]. They were able to isolate the cause of extreme familial hypercholesterolemia (FH), a Mendelian disorder, to mutations of a single gene encoding an LDL receptor. Moreover, they were able to identify the biochemical pathway which was affected by the mutation to create the disease condition. Statins target that pathway, making them useful not only to individuals suffering from FH, but also as an effective treatment for high LDL cholesterol in the general population.

29.3 Linkage Analysis

Historically, researchers have used **linkage analysis** to determine genetic variants which explain phenotypic variation. The goal is to determine which variants contribute to the observed pattern of phenotypic variation in a *pedigree*. Figure 29.3 shows an example pedigree in which squares are male individuals, circles are female individuals, couples and offspring are connected, and individuals in red have the trait of interest.

Linkage analysis relies on the biological insight that genetic variants are not independently inherited (as proposed by Mendel). Instead, meiotic recombination happens a limited number of times (roughly once per chromosome), so many variants *cosegregate* (are inherited together). This phenomenon is known as *linkage disequilibrium* (LD).

As the distance between two variants increases, the probability a recombination occurs between them increases. Thomas Hunt Morgan and Alfred Sturtevant developed this idea to produce **linkage maps** which could not only determine the order of genes on a chromosome, but also their relative distances to each other. The Morgan is the unit of genetic distance they proposed; loci separated by 1 centimorgan (cM) have 1 in

100 chance of being separated by a recombination. Unlinked loci have 50% chance of being separated by a recombination (they are separated if an odd number of recombinations happens between them). Since we usually do not know *a priori* which variants are causal, we instead use *genetic markers* which capture other variants due to LD. In 1980, David Botstein proposed using *single nucleotide polymorphisms* (SNPs), or mutations of a single base, as genetic markers in humans [4]. If a particular marker is in LD with the actual causal variant, then we will observe its pattern of inheritance contributing to the phenotypic variation in the pedigree and can narrow down our search.

The statistical foundations of linkage analysis were developed in the first part of the 20th century. Ronald Fisher proposed a genetic model which could reconcile Mendelian inheritance with continuous phenotypes such as height [8]. Newton Morton developed a statistical test called the **LOD score** (logarithm of odds) to test the hypothesis that the observed data results from linkage [23]. The null hypothesis of the test is that the *recombination fraction* (the probability a recombination occurs between two adjacent markers) $\theta = 1/2$ (no linkage) while the alternative hypothesis is that it is some smaller quantity. The LOD score is essentially a log-likelihood ratio which captures this statistical test:

$$\text{LOD} = \frac{\log(\text{likelihood of disease given linkage})}{\log(\text{likelihood of disease given no linkage})}$$

The algorithms for linkage analysis were developed in the latter part of the 20th century. There are two main classes of linkage analysis: *parametric* and *nonparametric* [29]. Parametric linkage analysis relies on a model (parameters) of the inheritance, frequencies, and penetrance of a particular variant. Let F be the set of founders (original ancestors) in the pedigree, let g_i be the genotype of individual i, let Φ_i be the phenotype of individual i, and let $f(i)$ and $m(i)$ be the father and mother of individual i. Then, the likelihood of observing the genotypes and phenotypes in the pedigree is:

$$L = \sum_{g_1} \dots \sum_{g_n} \prod_i \Pr(\Phi_i | g_i) \prod_{f \in F} \Pr(g_f) \prod_{i \notin F} \Pr(g_i | g_{f(i)}, g_{m(i)})$$

The time required to compute this likelihood is exponential in both the number of markers being considered and the number of individuals in the pedigree. However, Elston and Stewart gave an algorithm for more efficiently computing it assuming no inbreeding in the pedigree [7]. Their insight was that conditioned on parental genotypes, offspring are conditionally independent. In other words, we can treat the pedigree as a Bayesian network to more efficiently compute the joint probability distribution. Their algorithm scales linearly in the size of the pedigree, but exponentially in the number of markers.

There are several issues with parametric linkage analysis. First, individual markers may not be *informative* (give unambiguous information about inheritance). For example, homozygous parents or genotyping error could lead to uninformative markers. To get around this, we could type more markers, but the algorithm does not scale well with the number of markers. Second, coming up with model parameters for a Mendelian disorder is straightforward. However, doing the same for non-Mendelian disorders is non-trivial. Finally, estimates of LD between markers are not inherently supported.

Nonparametric linkage analysis does not require a genetic model. Instead, we first infer the inheritance pattern given the genotypes and the pedigree. We then determine whether the inheritance pattern can explain the phenotypic variation in the pedigree.

Lander and Green formulated an HMM to perform the first part of this analysis [17]. The states of this HMM are *inheritance vectors* which specify the result every meiosis in the pedigree. Each individual is represented by 2 bits (one for each parent). The value of each bit is 0 or 1 depending on which of the grand-parental alleles is inherited. Figure 29.4 shows an example of the representation of two individuals in an inheritance vector.

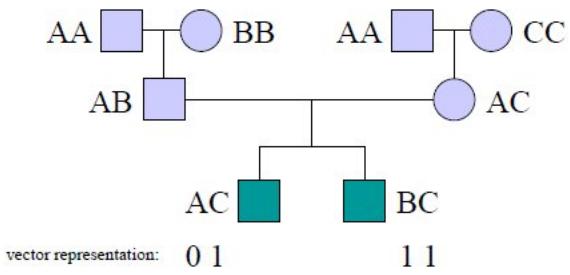


Figure 29.4: Representing a particular pattern of inheritance as an inheritance vector

Each step of the HMM corresponds to a marker; a transition in the HMM corresponds to some bits of the inheritance vector changing. This means the allele inherited from some meiosis changed, i.e. that a recombination occurred. The transition probabilities in the HMM are then a function of the recombination fraction between adjacent markers and the Hamming distance (the number of bits which differ, or the number of recombinations) between the two states. We can use the forward-backward algorithm to compute posterior probabilities on this HMM and infer the probability of every inheritance pattern for every marker.

This algorithm scales linearly in the number of markers, but exponentially in the size of the pedigree. The number of states in the HMM is exponential in the length of the inheritance vector, which is linear in the size of the pedigree. In general, the problem is known to be NP-hard (to the best of our knowledge, we cannot do better than an algorithm which scales exponentially in the input) [25]. However, the problem is important not only in this context, but also in the contexts of *haplotype inference* or *phasing* (assigning alleles to homologous chromosomes) and *genotype imputation* (inferring missing genotypes based on known genotypes). There have been many optimizations to make this analysis more tractable in practice [1, 9, 10, 12–15, 18, 20].

Linkage analysis identifies a broad genomic region which correlates with the trait of interest. To narrow down the region, we can use fine-resolution genetic maps of recombination breakpoints. We can then identify the affected gene and causal mutation by sequencing the region and testing for altered function.

29.4 Genome-wide Association Studies

Linkage analysis has proven to be highly effective in studying the genetic basis of Mendelian (single gene) diseases. In the past three decades, thousands of genes have been identified as contributing to Mendelian diseases. Figure 29.5 shows this explosion in published associations. We have identified the genetic basis of disease such as sickle cell anemia, cystic fibrosis, muscular dystrophy, and severe forms of common diseases such as diabetes and hypertension. For these diseases, mutations are severe and obvious; the environment, behavior, and chance have little effect.

However, most diseases (and many other traits of interest) are not Mendelian. These **complex traits** arise from the interactions of many genes and possibly the environment and behavior. A canonical complex trait is human height: it is highly heritable, but environmental factors can affect it. Recently, researchers have identified hundreds of variants which are associated with height [2, 22].

Linkage analysis is not a viable approach to find these variants. In the 1990s, researchers proposed a methodology called **genome-wide association** to systematically correlate markers with traits. These studies sample large pools of cases and controls, measure their genotypes at on the order of one million markers, and try to correlate variation in their genotypes with their variation in phenotype.

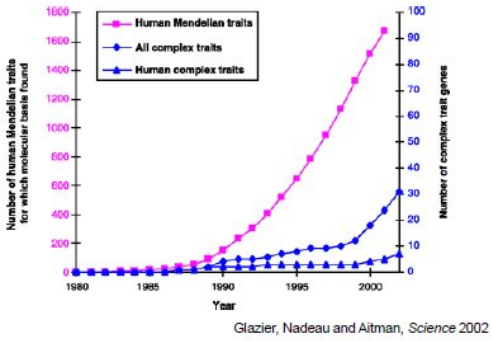


Figure 29.5: Discovery of genes for different disease types versus time

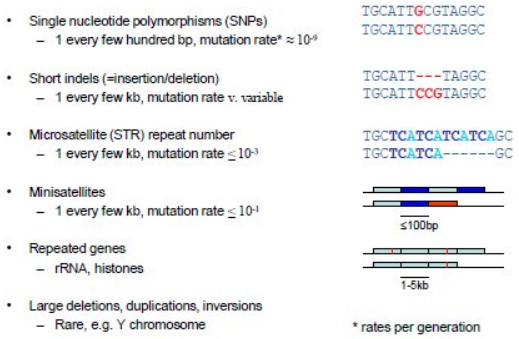


Figure 29.6: Different types of genetic variation

Genome-wide association studies (GWASs) are possible due to three advances. First, advances in our understanding of the genome and the creation of genomic resources. The key biological insight is the fact that humans are one of the least genetically diverse species. On the order of tens of millions of SNPs are shared between different human subpopulations. For any particular region of the genome, we observe only a limited number of **haplotypes** (allele combinations which are inherited together). Because of this high redundancy, we only need to measure a fraction of all the variants in the human genome in order to capture them all with LD. We can adapt the algorithms for inferring inheritance patterns in linkage analysis to impute genotypes for the markers which we did not genotype.

Genome resources allow us to carefully choose markers to measure and to make predictions based on markers which show statistically significant association. We now have the reference sequence of the human genome (allowing for alignments, genotype and SNP calling) and HapMap, a comprehensive catalog of SNPs in humans. We also have genome-wide annotations of genes and regulatory elements.

Second, advances in genotyping technology such as microarrays and high-throughput sequencing. Although there are many types of variation in the human genome (Figure 29.6 shows some examples), SNPs are the vast majority. They are also the easiest and cheapest to measure using these technologies. However, we still need to account for the other types of variants. Recently developed DNA microarrays can detect copy-number variation in addition to SNPs.

The third advance is a new expectation of collaboration between researchers. GWASs rely on large sample sizes to increase the *power* (probability of a true positive) of statistical tests. The explosion in the number of published GWASs has allowed for a new type of **meta-analysis** which combines the results of several GWASs for the same phenotype to make more powerful associations. Meta-analysis accounts for various

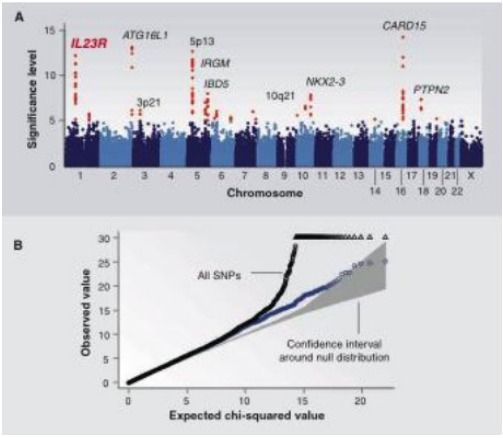


Figure 29.7: (A) Manhattan plot and (B) Q-Q plot for GWAS of Crohn's disease

technical and population-genetic biases in individual studies. Researchers who conduct GWASs are expected to collaborate with others who have conducted GWASs on the same trait in order to show replicability of results. By pooling together the data, we also have more confidence in the reported associations.

The main problem in conducting GWASs is eliminating confounding factors. First, genotyping error, which is common enough to require special treatment regardless of which technology we use. To account for such errors, we use thresholds on metrics like minor allele frequency and deviation from Hardy–Weinberg equilibrium and throw out SNPs which do not meet the criteria. Second, systematic genetic differences between human subpopulations. There are several methods to account for this **population substructure** such as genomic control [6], structured association [27], and principal component analysis [24, 26]. Third, covariates such as environmental and behavioral effects. We can account for these by including them in our statistical model.

The statistical analysis involved in GWAS is fairly straightforward. We assume the effect of each SNP is independent and additive to make the analysis tractable. For each SNP, we perform a hypothesis test whose null hypothesis is that the observed variation in the genotype at that SNP across the subjects does not correlate with the observed variation in the phenotype across the subjects. Because we perform one test for each SNP, we need to deal with the **multiple testing problem**. Each test has some probability of giving a false positive result, and as we increase the number of tests, the probability of getting a false positive in any of them increases. There are several methods to account for multiple testing such as Bonferroni correction and measures such as the false discovery rate [3] and the irreproducible discovery rate [19].

In addition to reporting SNPs which show the strongest associations, we typically also use *Manhattan plots* to show where these SNPs are located in the genome and *quantile-quantile (Q-Q) plots* to detect biases which have not been properly accounted for. A Manhattan plot is a scatter plot of log-transformed p-values against genomic position (concatenating the chromosomes). In Figure 29.7A, the points in red are those which meet the significance threshold. They are labeled with candidate genes which are close by.

A Q-Q plot is a scatter plot of log-transformed observed p-values against log-transformed expected p-values. We use uniform quantiles as the expected p-values: assuming there is no association, we expect p-values to be uniformly distributed. Deviation from the diagonal suggests p-values are more significant than would be expected. However, early and consistent deviation from the diagonal suggests too many p-values are too significant, i.e. there is some bias which is confounding the test. In Figure 29.7B, the plot shows observed test statistic against expected test statistic (which is equivalent). Considering all markers includes the Major Histocompatability Complex (MHC), which is the region associated with immune response. This

region has a unique LD structure which confounds the statistical analysis, as is clear from the deviation of the black points from the diagonal (the gray area). Throwing out the MHC removes much of this bias from the results (the blue points).

GWAS identifies markers which correlate with the trait of interest. However, each marker captures a neighborhood of SNPs which it is in LD with, making the problem of identifying the causal variant harder. Typically, the candidate gene for a marker is the one which is closest to it. From here, we have to do further study to identify the relevance of the variants which we identify. However, this remains a challenging problem for a few reasons:

- Regions of interest identified by association often implicate multiple genes
- Some of these associations are nowhere near any protein coding segments
- Linking these regions to underlying biological pathways is difficult

Our primary goal is to use these found associations to understand the biology of disease in an actionable manner, as this will help guide therapies in order to treat these diseases. This can be approached in one of two manners: the *bottom-up* approach, or the *top-down* approach.

The **bottom-up** approach is to investigate a particular gene that has a known association with a disease, and investigate it's biological importance within a cell. Kuballa et al.[16] were able to use this bottom-up approach to learn that a particular risk variant associated with Crohn's Disease leads to impairment of autophagy of certain pathogens. Furthermore, the authors were able to create a mouse model of the same risk variant found in humans. Identifying biological implications of risk variants at the cellular level and creating these models is invaluable as the models can be directly used to test new potential treatment compounds.

In contrast, the **top-down** approach involves looking at *all* known associations, and tries to link them to shared biological processes. This approach is based on the idea that many of the associated genes with a disease share relevant biological pathways. This is commonly done by taking existing networks like protein-protein interaction networks, and layering the associated genes on top of them. However, these resulting disease networks may not be significant due to bias in both the discovery of associations and the experimental bias of the data that the associations are being integrated with. This significance can be estimated by permuting the labels for the nodes in the network many times, and then computing how rare the level of connectivity is for the given disease network. This process is illustrated in Figure 29.8. As genes connected in the network should be co-expressed, it has been shown that these disease networks can be further validated from gene-expression profiling[11].

It is important to note GWAS captures more variants than linkage analysis. Linkage analysis identifies rare variants which have negative effects. GWAS can also identify these variants, but in addition can identify rare variants which have protective effects. Linkage analysis cannot identify these variants because they are anti-correlated with disease status. More importantly, GWAS can identify common variants with smaller effect sizes. Linkage analysis relies on the assumption that a single variant explains the disease. But this assumption does not hold for complex traits such as disease. Instead, we need to consider many markers in order to explain the genetic basis of these traits.

We have learned several lessons from GWAS. First, fewer than one-third of reported associations are coding or obviously functional variants. Second, only some fraction of associated non-coding variants are significantly associated to expression level of a nearby gene. Third, many are associated to regions with no nearby coding gene. Finally, the majority of reported variants are associated to multiple autoimmune or inflammatory diseases. These revelations indicate that there are still many mysteries lurking in the genome waiting to be discovered.

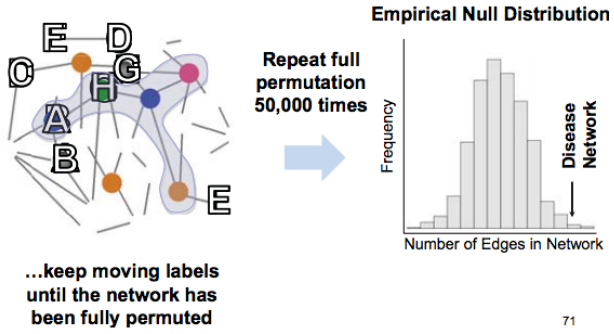


Figure 29.8: Evaluating Disease Network Significance

29.5 Current Research Directions

GWAS helps to identify loci that are associated with a particular disease, but they do not point out specific causal alleles related to these diseases of interest. Advances in **next gen sequencing** (NGS) have made sequencing an individuals genome a much less costly and time consuming task. However, getting from the genome to any casual alleles can be a difficult task. Any given individual may have hundreds of rare variants in their genome that may be causing loss of function in a gene[21], however, locating this variants and associating them with a particular disease is challenging. However, by performing base pair level sequencing of GWAS confirmed loci, specific alleles relating to disease have emerged[28].

29.6 Further Reading

29.7 Tools and Techniques

29.8 What Have We Learned?

In the past several decades, we have made huge advances in developing techniques to investigate the genetic basis of disease. Historically, we have used linkage analysis to find causal variants for Mendelian disease with great success. More recently, we have used genome-wide association studies to begin investigating more complex traits with some success. However, more work is needed in developing methods to interpret these GWAS and identifying causal variants and their role in disease mechanism. Improving our understanding of the genetic basis of disease will us develop more effective diagnoses and treatments.

Bibliography

[1] G.R. Abeçasis, S.S. Cherny, W.O. Cookson, and L.R. Cardon. Merlin—rapid analysis of dense genetic maps using sparse gene flow trees. *Nature Genetics*, 30(1):97–101, 2002.

[2] H.L. Allen et al. Hundreds of variants clustered in genomic loci and biological pathways affect human height. *Nature*, 467(7317):832–838, 2010.

[3] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, 57:289–300, 1995.

[4] D. Botstein, R.L. White, M. Skolnick, and R.W. Davis. Construction of a genetic linkage map in man using restriction fragment length polymorphisms. *American Journal of Human Genetics*, 32:314–331, 1980.

[5] M.S. Brown and J.L. Goldstein. A receptor-mediated pathway for cholesterol homeostasis. *Science*, 232(4746):34–47, 1986.

[6] B. Devlin and K. Roeder. Genomic control for association studies. *Biometrics*, 55:997–1004, 1999.

[7] R.C. Elston and J. Stewart. A general model for the genetic analysis of pedigree data. *Human Heredity*, 21:"523–542", 1971.

[8] Sir R.A. Fisher. The correlation between relatives on the supposition of Mendelian inheritance. *Transactions of the Royal Society of Edinburgh*, 52:399–433, 1918.

[9] D.F. Gudbjartsson, K. Jonasson, M.L. Frigge, and A. Kong. Allegro, a new computer program for multipoint linkage analysis. *Nature Genetics*, 25(1):12–13, 2000.

[10] D.F Gudbjartsson, T. Thorvaldsson, A. Kong, G. Gunnarsson, and A. Ingolfsdottir. Allegro version 2. *Nature Genetics*, 37(10):1015–1016, 2005.

[11] X. Hu, H. Kim, E. Stahl, R. Plenge, M. Daly, and S. Raychaudhuri. Integrating autoimmune risk loci with gene-expression data identifies specific pathogenic immune cell subsets. *The American Journal of Human Genetics*, 89(4):496–506, 2011.

[12] R.M. Idury and R.C. Elston. A faster and more general hidden markov model algorithm for multipoint likelihood calculations. *Human Heredity*, 47:197–202, 1997.

[13] A. Ingolfsdottir and D. Gudbjartsson. Genetic linkage analysis algorithms and their implementation. In Corrado Priami, Emanuela Merelli, Pablo Gonzalez, and Andrea Omicini, editors, *Transactions on Computational Systems Biology III*, volume 3737 of *Lecture Notes in Computer Science*, pages 123–144. Springer Berlin / Heidelberg, 2005.

[14] L. Kruglyak, M.J. Daly, M.P. Reeve-Daly, and E.S. Lander. Parametric and nonparametric linkage analysis: a unified multipoint approach. *American Journal of Human Genetics*, 58:1347–1363, 1996.

[15] L. Kruglyak and E.S. Lander. Faster multipoint linkage analysis using fourier transforms. *Journal of Computational Biology*, 5:1–7, 1998.

[16] P. Kuballa, A. Huett, J.D. Rioux, M.J. Daly, and R.J. Xavier. Impaired autophagy of an intracellular pathogen induced by a crohn's disease associated atg16l1 variant. *PLoS One*, 3(10):e3391, 2008.

[17] E.S. Lander and P. Green. Construction of multilocus genetic linkage maps in humans. *Proceedings of the National Academy of Sciences*, 84(8):2363–2367, 1987.

[18] E.S. Lander, P. Green, J. Abrahamson, A. Barlow, M.J. Daly, S.E. Lincoln, and L. Newburg. Mapmaker: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics*, 1(2):174–181, 1987.

[19] Q. Li, J.B. Brown, H. Huang, and P.J. Bickel. Measuring reproducibility of high-throughput experiments. *Annals of Applied Statistics*, 5:1752–1797, 2011.

[20] E.Y. Liu, Q. Zhang, L. McMillan, F.P. de Villena, and W. Wang. Efficient genome ancestry inference in complex pedigrees with inbreeding. *Bioinformatics*, 26(12):i199–i207, 2010.

[21] D.G. MacArthur, S. Balasubramanian, A. Frankish, N. Huang, J. Morris, K. Walter, L. Jostins, L. Habegger, J.K. Pickrell, S.B. Montgomery, et al. A systematic survey of loss-of-function variants in human protein-coding genes. *Science*, 335(6070):823–828, 2012.

[22] B.P. McEvoy and P.M. Visscher. Genetics of human height. *Economics & Human Biology*, 7(3):294 – 306, 2009.

[23] N.E. Morton. Sequential tests for the detection of linkage. *The American Journal of Human Genetics*, 7(3):277–318, 1955.

[24] N. Patterson, A. Price, and D. Reich. Population structure and eigenanalysis. *PLoS Genetics*, 2:e190, 2006.

[25] A. Piccolboni and D. Gusfield. On the complexity of fundamental computational problems in pedigree analysis. *Journal of Computational Biology*, 10:763–773, October 2003.

[26] A. Price et al. Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, 38:904–909, 2006.

[27] J. Pritchard, M. Stephens, N. Rosenberg, and P. Donnelly. Association mapping in structured populations. *American Journal of Human Genetics*, 67:170–181, 2000.

[28] M.A. Rivas, M. Beaudoin, A. Gardet, C. Stevens, Y. Sharma, C.K. Zhang, G. Boucher, S. Ripke, D. Ellinghaus, N. Burtt, et al. Deep resequencing of gwas loci identifies independent rare variants associated with inflammatory bowel disease. *Nature genetics*, 2011.

[29] T. Strachan and A.P. Read. *Human Molecular Genetics*. Wiley-Liss, New York, 2 edition, 1999.

CHAPTER

THIRTY

MISSING HERETIBILITY

TODO: missing *@scribe: add author*

30.1 Introduction

30.2 Current Research Directions

30.3 Further Reading

30.4 Tools and Techniques

30.5 What Have We Learned?

Bibliography

CHAPTER

THIRTYONE

PERSONAL GENOMES, SYNTHETIC GENOMES, COMPUTNG IN C VS. SI

Guest Lecture by George Church
Scribed by Lawson Wong (2011)

31.1 Introduction

George Church discussed a variety of topics that have motivated his past and present research. He first discussed about reading and writing genomes, including his own involvement in the development of sequencing and the Human Genome Project. In that latter half, he discussed about his more recent endeavor, the Personal Genome Project, which he initiated in 2005.

31.2 Reading and Writing Genomes

As a motivation, consider the following question: Is there any technology that is not biologically motivated or inspired? Biology and our observations of it influence our lives pervasively. For example, within the energy sector, biomass and bioenergy has always existed and is increasingly becoming the focus of attention. Even in telecommunications, the potential of quantum-level molecular computing is promising, and is expected to be a major player in the future.

Church has been involved in molecular computing in his own research, and claims that once harnessed, it has great advantages over their current silicon counterparts. For example, molecular computing can provide at least 10% greater efficiency per Joule in computation. More profound perhaps is its potential effect on data storage. Current data storage media (magnetic disk, solid-state drives, etc.) is much less (billions times) dense than DNA. The limitation of DNA as data storage is that it has a high error rate. Church is currently involved in a project exploring reliable storage through the use of error correction and other techniques.

In a 2009 Nature Biotechnology review article [1], Church explores the potential for efficient methods to read and write to DNA. He observes that in the past decade there has been a 10× exponential curve in both sequencing and oligo synthesis, with double-stranded synthesis lagging behind but steadily increasing. Compared to the 1.5× exponential curve for VLSI (Moore's Law), the increase on the biological side is more dramatic, and there is no theoretical argument yet for why the trend should taper off. In summary, there is great potential for genome synthesis and engineering.

Did You Know?

George Church was an early pioneer of genome sequencing. In 1978, Church was able to sequence plasmids at $10 per base. By 1984, together with Walter Gilbert, he developed the first direct genomic sequencing method [3]. With this breakthrough, he helped initiate the Human Genome Project in 1984. This proposal aimed to sequence an entire human haploid genome at $1 per base, requiring a total budget of $3 billion. This quickly played out into the well-known race between Celera and UCSC-Broad-Sanger. Although the latter barely won in the end, their sequence had many errors and gaps, whereas Celera's version was much higher quality. Celera initially planned on releasing the genome in 50 kb fragments, which researchers could perform alignments on, much like BLAST. Church once approached Celera's founder, Craig Venter, and received a promise to obtain the entire genome on DVD after release. However, questioning the promise, Church decided instead to download the genome directly from Celera by taking advantage of the short fragment releases. Using automated crawl and download scripts, Church managed to download the entire genome in 50 kb fragments within three days!

31.3 Personal Genomes

In 2005, George Church initiated the Personal Genome Project [2]. Now that sequencing costs have rapidly decreased to the point that we can currently get the entire diploid human genome for $4000 (compare to $3 billion for a haploid human genome in the Human Genome Project), personal genome and sequence information is becoming increasingly affordable.

One important application for this information is in personalized medicine. Although many diseases are still complicated to predict, diagnose, and study, we currently already have a small list of diseases that are highly predictable from genome data. Examples include phenylketonuria (PKU), BRCA-mutation-related breast cancer, and hypertrophic cardiomyopathy (HCM). Many of these and similar diseases are uncertain (sudden onset without warning symptoms) and not normally checked for (due to their relative rareness). As such, they are particularly suitable as targets for personalized medicine by personal genomes, because genomic data provide accurate information that otherwise cannot be obtained. Already, there are over 2500 diseases (due to $\sim$ 6000 genes) that are highly predictable and medically actionable, and companies such as 23andMe are exploring these opportunities.

As a final remark on the subject, Church remarked on some of his personal philosophy regarding personalized medicine. He finds many people reluctant to obtain their genomic information, and attributes this to a negative view among the general public toward GWAS and personalized medicine. He thinks that the media focuses too much on the failure of GWAS. The long-running argument against personalized medicine is that we should focus first on common diseases and variants before studying rare events. Church counterargues that in fact there is no such thing as a common disease. Phenomena such as high blood pressure or high cholesterol only count as symptoms; many 'common diseases' such as heart disease and cancer have many subtypes and finer categories. All along, lumping these diseases into one large category only has the benefit of teaching medical students and to sell pharmaceuticals (e.g., statins, which have fared well commercially but only benfit very few). Church argues that lumping implies a loss of statistical power, and is only useful if it is actually meaningful. Ultimately, everyone dies due to their own constellation of genes and diseases,

so Church sees that splitting (personalized genomics) is the way to proceed.

Personal genomics provide information for planning and research. As a business model, it is analogous to an insurance policy, which provides risk management. As an additional benefit however, the information received allows for early detection, and consequences may even be avoidable. Access to genomic information allows one to make more informed decisions.

31.4 Current Research Directions

31.5 Further Reading

Personal Genome Project: http://www.personalgenomes.org/

31.6 Tools and Techniques

31.7 What Have We Learned?

Bibliography

[1] Peter A. Carr and George M. Church. Genome engineering. *Nature biotechnology*, 27(12):1151–1162, December 2009.

[2] G. M. Church. The Personal Genome Project. *Molecular Systems Biology*, 1(1):msb4100040–E1–msb4100040–E3, December 2005.

[3] G. M. Church and W. Gilbert. Genomic sequencing. *Proceedings of the National Academy of Sciences of the United States of America*, 81(7):1991–1995, April 1984.

CHAPTER

THIRTYTWO

PERSONAL GENOMICS

TODO: missing *@scribe: add author*

32.1 Introduction

32.2 Current Research Directions

32.3 Further Reading

32.4 Tools and Techniques

32.5 What Have We Learned?

Bibliography