1 Introduction

1.1 Why Use R?

- Widely-used (ever-increasingly so in political science)
- Free
- Power and flexibility
- Graphical capabilities
- Pedagogical usefulness
- Learning curve is initially steep, but it gets easier later

1.2 Installing R

1. Go to http://www.r-project.org
2. Follow the link under the Download section on the left hand side of the page.
3. Make sure to select an appropriate mirror (the closer the faster) and operating system.
4. Once the download is finished, you can install R in the same way you install other software.

1.3 Using R

1.3.1 Typing Directly

- R as a calculator
- <- or = to assign a name to an object (<- is recommended)
- # for comments

```r
> apple <- 7
> apple
[1] 7
```
> ## Can use = but not recommended
> orange = 12
> orange
>
> [1] 12

- Hello ≠ hello ≠ HELLO

> obama <- "president" # character string
> Obama # generates error

- Variable names cannot begin with numbers

- One command per line or use ;

> pol572 <- "Quant II"; enrollment <- 26
> pol572

> enrollment
>
> [1] 26

- Space between syntaxes will be ignored, but not in ""

> pol.572 <- "second in quant sequence"
> pol.572

> [1] "second in quant sequence"

- Can use <up> and <down> to navigate through previously entered commands

- An example: calculate the amount of Princeton's endowment the university could spend for each student without harming the principle.

> 15787000000 / (4918+2416) # Endowment divided by the total number of students

> [1] 2152577

> endow.ps <- 15.787*(10^9) / (4918+2416) # Stores the result as `endow.ps`
> yield.ps <- endow.ps*0.08 # Creates new object `yield.ps`
> yield.ps

> [1] 172206.2
1.3.2 Using a Text Editor

- Almost always want to use a text editor
- Efficiency, Replicability
- Built-in R editor; what we’ll use in the camp
- Notepad, TextPad, TextEdit, etc.
- More fancy editors: WinEdt (http://www.winedt.com) for Windows and Aquamacs (http://aquamacs.org) for Mac OS X
- Send code from R editor to console by:
  - Copy & Paste
  - Highlight and press <Ctrl + R> on Windows
  - Highlight and press <Apple + Return> on Mac
- An example: Repeat the endowment calculation above using the R editor.

1.4 Numeric Vectors

A vector is simply a string of numeric values connected in a specific order.

1.4.1 Creating, Indexing and Combining

- Use c() to enter a data vector
- Use square brackets ([ ]) to access elements of a vector
- c() can also combine more than one vectors

```r
> endow <- c(9928, 11207, 13045, 15000) #Princeton endowment 2004-2007
> endow
[1]  9928 11207 13045 15000

> endow[2]  # Use square brackets to access elements of a vector
[1] 11207

> endow[c(2, 4)]
[1] 11207 15000

> endow[4] <- 15787
> endow
[1]  9928 11207 13045 15787

> endow[-1]
```
> endow <- endow / 1000
> endow

> endow.old <- c(8.398, 8.359, 8.320, 8.730) #Years 2000-2003
> endow <- c(endow.old, endow) #combine vectors
> endow

1.4.2 Manipulating

- Named vectors can be manipulated like numbers

> endow.penn <- c(3.201, 3.382, 3.393, 3.547, 4.019, 4.370, 5.313, 6.635) #UPenn endowment
> endow.penn

> ratio <- endow / endow.penn
> ratio
[1] 2.623555 2.471614 2.452107 2.461235 2.470266 2.564531 2.455298 2.379352

1.4.3 Basic Functions

- General format of a function: output <- functionname(input)
- More than one input: output <- functionname(input1, input2, ...)
- Inputs are often called arguments
- The colon operator (:) creates a simple sequence
- Use seq() for more complex sequences

> 1:10
[1] 1 2 3 4 5 6 7 8 9 10

> seq(from = 2, to = 10, by = 2)
[1] 2 4 6 8 10

- names() gives named entries to a vector

> names(ratio) <- 2000:2007
> ratio
Simple calculations: \texttt{min()}, \texttt{max()}, \texttt{range()}, \texttt{length()}, \texttt{sum()} and \texttt{mean()}

\begin{verbatim}
> min(ratio)  # Shows the minimum value of our vector
[1] 2.379352

> max(ratio)  # Same, but for the maximum value
[1] 2.623555

> range(ratio)
[1] 2.379352 2.623555

> length(ratio)  # Tells us the number of values in our vector
[1] 8

> ratio[length(ratio)]

2007
2.379352

> sum(ratio)  # Sum of all values in the vector
[1] 19.87796

> sum(ratio)/length(ratio)  # Long way to calculate the mean
[1] 2.484745

> mean(ratio)  # Easier
[1] 2.484745
\end{verbatim}

1.5 Matrices

1.5.1 Creating, Indexing and Combining

\begin{itemize}
  \item R can also handle \texttt{matrices}, two-dimensional array of numbers.
  \item The \texttt{matrix(x, N, K)} function reshapes a vector \texttt{x} into an \(N \times K\) matrix.
  \item By default, a matrix is (counterintuitively) filled by columns. Setting the \texttt{byrow} argument to \texttt{TRUE} (or just \texttt{T}) change this behavior.
\end{itemize}

\begin{verbatim}
> a <- c(1,2,3,4)
> X <- matrix(a, 2, 2)
> X
\end{verbatim}
\[
\begin{bmatrix}
[,1] & [,2] \\
[1,] & 1 & 3 \\
[2,] & 2 & 4 \\
\end{bmatrix}
\]

\[Y <- \text{matrix}(a, 2, 2, \text{byrow}=\text{TRUE})\]
\[Y \# \text{Notice how } X \text{ and } Y \text{ differ}\]

\[
\begin{bmatrix}
[,1] & [,2] \\
[1,] & 1 & 2 \\
[2,] & 3 & 4 \\
\end{bmatrix}
\]

- A matrix can also be built up from other vectors/matrices.
- The \text{cbind}(x,y) and \text{rbind}(x,y) functions combine \text{x} and \text{y} by columns and rows, respectively.

\[b <- \text{c}(5,6,7,8)\]
\[V <- \text{cbind}(a,b)\]
\[V\]

\[
a b \\
[1,] & 1 & 5 \\
[2,] & 2 & 6 \\
[3,] & 3 & 7 \\
[4,] & 4 & 8 \\
\]

\[W <- \text{rbind}(a,b)\]
\[W\]

\[
\begin{bmatrix}
a & 1 & 2 & 3 & 4 \\
b & 5 & 6 & 7 & 8 \\
\end{bmatrix}
\]

- The \text{nrow}(X) function returns the number of rows in \text{X}; \text{ncol}(X) returns the number of columns.
- The \text{dim}(X) function reports both at the same time.
- Use square brackets and a comma ([,]) to specify rows and columns.

\[nrow(V)\]
\[\text{[1]} \ 4\]
\[ncol(V)\]
\[\text{[1]} \ 2\]
\[dim(V)\]
\[\text{[1]} \ 4 \ 2\]
\[V[1,] \# \text{returns the first row}\]
\texttt{a b}
\texttt{1 5}

\texttt{\textgreater{} V[3,2] \# returns the (3,2) element}

\texttt{b}
\texttt{7}

1.5.2 Manipulating

- The addition operator (\texttt{+}) works on matrices in the same way as usual

\texttt{\textgreater{} X + Y}

\begin{verbatim}
[,1] [,2]
[1,] 2 5
[2,] 5 8
\end{verbatim}

- However, the multiplication (\texttt{\textast{}}) works element by element

- Instead, the \texttt{\textperc{}} operator is used for usual matrix multiplication

\texttt{\textgreater{} X \textperc{} Y}

\begin{verbatim}
[,1] [,2]
[1,] 1 6
[2,] 6 16
\end{verbatim}

\texttt{\textgreater{} X \textperc{}} Y

\begin{verbatim}
[,1] [,2]
[1,] 10 14
[2,] 14 20
\end{verbatim}

- Other matrix operations include:
  - \texttt{t(X)} (transpose, \textit{X}'
  - \texttt{det(X)} (determinant, \textit{|X|})
  - \texttt{solve(X)} (inversion, \textit{X}^{-1})
  - \texttt{crossprod(X,Y)} (cross product, \textit{X}'Y)
  - \texttt{kronecker(X,Y)} or \texttt{X \textperc{}} (Kronecker product, \textit{X} \textcircled{Y})
1.6 Object Class

- An object belongs to a certain class (e.g. numeric, character, function)
- The function class() returns the class of an object
- The function summary() to get a summary of an object

```r
> endowment <- c(9928, 11207, 13045, 15787)
> class(endowment)

[1] "numeric"

> summary(endowment)

     Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
       9928 10890 12130   12490 13730   15790

> class(schools)

[1] "character"

> summary(schools)  # Not so useful when dealing with "character" vectors

     Length Class  Mode
       6 character character

- factor is more useful for categorical variables than character
- The function as.factor() coerces an object into a factor (as.numeric(), as.character())
- The function levels() returns the categories of a factor

```r
> regions <- c("Africa", "Africa", "Asia", "Asia", "Africa", "Middle East")
> regions

[1] "Africa" "Africa"  "Asia"    "Asia"    "Africa"    "Middle East"
[6] "Middle East"

> class(regions)

[1] "character"

> regions <- as.factor(regions)
> regions

[1] Africa Africa Asia Asia Africa Middle East
Levels: Africa Asia Middle East

> class(regions)

[1] "factor"
> levels(regions)  # Displays all possible categories for our factor
[1] "Africa"     "Asia"       "Middle East"

> summary(regions)  # Useful output for factors

<table>
<thead>
<tr>
<th></th>
<th>Africa</th>
<th>Asia</th>
<th>Middle East</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

1.7 Workspace

- The workspace is the sandbox where R temporarily saves everything you’ve created or loaded into R.
- The functions `objects()` and `ls()` list all objects in the workspace.
- To remove an object from the workspace, use `remove()` or `rm()`.
- You can remove multiple objects via `rm(object1, object2, ...)`

   > objects()

   [1] "a"    "apple" "b"    "endow" "endowment"
   [6] "endow.old" "endow.penn" "endow.ps" "enrollment" "orange"
   [11] "pol1572" "pol.572" "ratio" "regions" "schools"
   [16] "V"    "W"    "X"    "Y"    "yield.ps"

   > rm(schools)

   > ls()

   [1] "a"    "apple" "b"    "endow" "endowment"
   [6] "endow.old" "endow.penn" "endow.ps" "enrollment" "orange"
   [11] "pol1572" "pol.572" "ratio" "regions" "V"
   [16] "W"    "X"    "Y"    "yield.ps"

1.8 Saving Objects and Workspace

- Use `save()` to save an object or multiple objects (or go to the pulldown menu under File: “Save Workspace...”)
- The file name should be `xxx.RData`

   > save(endowment, file = "H:/endowment.RData")
   > save(endowment, regions, file = "H:/endowment.RData")  # save multiple objects

- If you want to save all objects (i.e., the entire workspace), then use `save.image()`

   > save.image("H:/Handout1.RData")
• The working directory is where R loads and saves your files.

• For example, `save(endowment, file = "endowment.RData")` will save `endowment.RData` in the working directory.

• When you close down R, you will be asked if you’d like to save your current workspace. You normally want to choose NO.

• If you choose YES, R will save your files in the working directory as “.RData” (empty file name + extension).

• `getwd()` displays the working directory
  ```r
  > getwd()
  [1] "/home/teppei/Documents/imai/software/handouts"
  ```

• `setwd()` changes the working directory (or go to the pulldown menu)
  ```r
  > setwd("H:/") # Start saving to H drive
  ```

• Change the working directory to an appropriate directory at the beginning of your code.

1.9 Using Additional Packages

• R can be extended via **packages**, which contains additional functions and example data.

• Many packages are already available, and the library is being expanded by statisticians and methodologists.

• Packages can be installed directly from the internet using `install.packages("xxx")` where `xxx` is the package name.
  ```r
  > install.package("mvtnorm") # a package for multivariate normal distribution
  ```

• To use an installed package, type `library(xxx)`.
  ```r
  > # Draws from the bivariate standard normal distribution
  > rmvnorm(1, mean=c(0,0), sigma=matrix(c(1,0,0,1), nrow=2)) # generates error
  > library(mvtnorm)
  > rmvnorm(1, mean=c(0,0), sigma=matrix(c(1,0,0,1), nrow=2))
    [,1] [,2]
   [1,] 0.2674600 1.082827
  ```
<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
<th>Pre-Installed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>boot</td>
<td>Functions for bootstrap analysis</td>
<td>Yes</td>
</tr>
<tr>
<td>foreign</td>
<td>Read data files created in other softwares, e.g., Stata, SAS, SPSS.</td>
<td>Yes</td>
</tr>
<tr>
<td>MASS</td>
<td>A large collection of useful functions by Venables &amp; Lipley (2002)</td>
<td>Yes</td>
</tr>
<tr>
<td>MCMCpack</td>
<td>Bayesian analysis via Markov chain Monte Carlo</td>
<td>No</td>
</tr>
<tr>
<td>mvtnorm</td>
<td>Multivariate normal and t distributions.</td>
<td>No</td>
</tr>
<tr>
<td>Zelig</td>
<td>Easy-to-use frontend for a range of statistical models by Imai, King &amp; Lau</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 1: List of Add-on Packages Often Used in Political Science.

1.10 Getting R Help

- Use `help()` or `?` to look up help on a function
- `help.search("xxx")` to look for functions that mention the word `xxx`

```r
> help("save")
> ?save    # Same thing
> ?read.table
> help.search("save")
```

- Many resources on the web (e.g., search, mailing list at [http://cran.r-project.org](http://cran.r-project.org))

2 Working with Data

2.1 Loading Workspace and Data

- If you have the file named “.RData” in the working directory, it will get automatically loaded when you start R in that directory
- `load()` for a previously saved R object/workspace (`xxx.RData`)
- `read.table()` for a simple text (ASCII) file (`xxx.txt`, etc.)
- `read.csv()` for a comma-delimited file (`xxx.csv`)
- `read.delim()` for a tab-delimited file (`xxx.tab`, etc.)

- Use `header = T` (for “TRUE”) for a file with variable name headers in the first row
- `read.table()` defaults to `header = T`; others to `F` (= “FALSE”)

- The object class for datasets is `data.frame`

```r
> load("Africa.RData")    # Looks for the file in our working directory
> Africa <- read.table("Africa.txt", header = TRUE)
> Africa <- read.csv("Africa.csv", header = TRUE)
> class(Africa)

[1] "data.frame"
```
We can also pull data straight from the web

```r
> primates <- "http://www.hopkinsmedicine.org/FAE/primate_structural_properties.txt"
> hop <- read.delim(primates, header=T)
```

### 2.2 Reading Data Created in Other Softwares

- The `foreign` package must be loaded to read files created in other softwares.
- `read.dta()` reads a Stata data file (`xxx.dta`)
- `read.spss()` reads a SPSS data file (`xxx.sav`)

```r
> library(foreign)
> Humph <- read.dta("Humph.dta") # Data on African Countries by Humphreys
> BPCD <- read.spss("BPCD.sav") # British Parliament Constituency Data by Norris
```

- To read an Excel spreadsheet file (`xxx.xls`), the simplest way is to open it in Excel, save it as a `.csv` file and read it with `read.csv()`.
  (Note: The `gdata` package contains the `read.xls()` function, which can directly read an Excel file. However, you need to have the Perl language installed to get it to work.)

### 2.3 Looking at Data

- `names()` gives variable names in the data frame
- `nrow()`, `ncol()` and `dim()` work for data frames in the same as they do for matrices
- Also like matrices, data frames can be indexed by `[ , ]`
- Use `$` to access an individual variable
- `summary()` works for data frames too

```r
> names(Africa)
[1] "Country" "GDP" "GDP.pc" "HDI" "Region"

> dim(Africa)
[1] 52 5

> Africa[1:4,] # Display the first four rows, and all columns
    Country   GDP   GDP.pc  HDI Region
   1 Algeria 298448 8649 0.73  Northern Africa
   2 Angola   91825  5463 0.45   Central Africa
   3 Benin   12217  1507 0.44   Western Africa
   4 Botswana 28454 18402 0.65    Southern Africa

> Africa[3:8, c(2,4)] # Display rows 3-8, columns 2 and 4
> GDP HDI
3 12217 0.44
4 28454 0.65
5 22132 0.37
6 5913 0.41
7 45777 0.53
8 4271 0.74

> Africa$HDI  # Displays only the HDI values

[1] 0.73 0.45 0.44 0.65 0.37 0.41 0.53 0.74 0.38 0.39 0.56 0.43 0.41 0.52 0.71
[16] 0.64 0.48 0.41 0.68 0.50 0.55 0.46 0.37 0.52 0.55 0.33 0.82 0.53 0.44 0.38
[31] 0.55 0.80 0.65 0.38 0.65 0.37 0.47 0.55 0.45 0.65 0.50 0.84 0.34 0.67 0.53
[46] 0.55 0.47 0.51 0.77 0.51 0.43 0.51

> summary(Africa)

    Country GDP  GDP.pc HDI
Algeria : 1 Min. : 616 Min. : 500 Min. :0.3300
Angola : 1 1st Qu.: 6178 1st Qu.: 1366 1st Qu.:0.4300
Benin : 1 Median :19654 Median : 2162 Median :0.5100
Botswana: 1 Mean : 61118 Mean : 4616 Mean :0.5294
Burkina Faso: 1 3rd Qu.: 55461 3rd Qu.: 5569 3rd Qu.:0.6425
Burundi : 1 Max. :703709 Max. :23294 Max. :0.8400
(Other) :46

    Region             
Central Africa : 9
Eastern Africa :16
Northern Africa: 6
Southern Africa: 5
Western Africa :16

2.4 Logical, Conditional Statements and Subsetting

- Logical operators: <, <=, >, >=, == and !=

- an object class of logical (i.e., TRUE or FALSE) is returned

- “==” versus “=”

- Works with vectors too

> 4 > 3
[1] TRUE

> "Hello" == "hello"
[1] FALSE
> x <- 4 > 3
> x

[1] TRUE

> class(x)

[1] "logical"

> x <- c(3, 2, 1, -2, -1)
> x >= 2

[1] TRUE TRUE FALSE FALSE FALSE

> x > 0 & x <= 2

[1] FALSE TRUE TRUE FALSE FALSE

> x > 0 | x <= 2

[1] TRUE TRUE TRUE TRUE TRUE

- Combine logical statements with & (and) and | (or)

> x > 0 & x <= 2

[1] FALSE TRUE TRUE FALSE FALSE

> x > 0 | x <= 2

[1] TRUE TRUE TRUE TRUE TRUE

- Use logical statements to subset the data. Two important arguments of `subset()` are `subset` (for observations) and `select` (for variables)

> Africa[Africa$Region == "Northern Africa",]

<table>
<thead>
<tr>
<th>Country</th>
<th>GDP</th>
<th>GDP.pc</th>
<th>HDI</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algeria</td>
<td>298448</td>
<td>8649</td>
<td>0.73</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>15 Egypt</td>
<td>423464</td>
<td>5643</td>
<td>0.71</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>27 Libya</td>
<td>93402</td>
<td>15041</td>
<td>0.82</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>33 Morocco</td>
<td>198785</td>
<td>6406</td>
<td>0.65</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>45 Sudan</td>
<td>129447</td>
<td>3395</td>
<td>0.53</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>49 Tunisia</td>
<td>107185</td>
<td>10269</td>
<td>0.77</td>
<td>Northern Africa</td>
</tr>
</tbody>
</table>

> subset(Africa, subset = (Region == "Northern Africa"))

<table>
<thead>
<tr>
<th>Country</th>
<th>GDP</th>
<th>GDP.pc</th>
<th>HDI</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algeria</td>
<td>298448</td>
<td>8649</td>
<td>0.73</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>15 Egypt</td>
<td>423464</td>
<td>5643</td>
<td>0.71</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>27 Libya</td>
<td>93402</td>
<td>15041</td>
<td>0.82</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>33 Morocco</td>
<td>198785</td>
<td>6406</td>
<td>0.65</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>45 Sudan</td>
<td>129447</td>
<td>3395</td>
<td>0.53</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>49 Tunisia</td>
<td>107185</td>
<td>10269</td>
<td>0.77</td>
<td>Northern Africa</td>
</tr>
</tbody>
</table>

> Africa[(Africa$GDP.pc >= 8000) & (Africa$Region != "Northern Africa"), ]

<table>
<thead>
<tr>
<th>Country</th>
<th>GDP</th>
<th>GDP.pc</th>
<th>HDI</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algeria</td>
<td>298448</td>
<td>8649</td>
<td>0.73</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>15 Egypt</td>
<td>423464</td>
<td>5643</td>
<td>0.71</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>27 Libya</td>
<td>93402</td>
<td>15041</td>
<td>0.82</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>33 Morocco</td>
<td>198785</td>
<td>6406</td>
<td>0.65</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>45 Sudan</td>
<td>129447</td>
<td>3395</td>
<td>0.53</td>
<td>Northern Africa</td>
</tr>
<tr>
<td>49 Tunisia</td>
<td>107185</td>
<td>10269</td>
<td>0.77</td>
<td>Northern Africa</td>
</tr>
</tbody>
</table>
We can add up the number of TRUE statements using the `sum()` command

```r
> x
[1] 3 2 1 -2 -1
> x >= 2
[1] TRUE TRUE FALSE FALSE FALSE FALSE
> sum(x>=2)  # Adds up the number of TRUE statements
[1] 2
```

Conditional Statements evaluate a logical statement, then perform an action

- `ifelse(X, Y, Z)` performs `Y` if the statement `X` is true; performs `Z` if `X` is false
2.5 Working with Missing Values

- Missing values in your dataset can cause you trouble, and sometimes mess up your analysis completely.
- R uses NA ("not available") to indicate missing values.
- When a calculation involves NAs, it often behaves in a rather counterintuitive manner. Examples:
  > x <- c(0, 1, NA)
  > x > 0 # Returns "NA" for NA, rather than "FALSE"
  [1] FALSE TRUE NA
  > x == NA # Returns "NA" for everything
  [1] NA NA NA
- The is.na() function tells whether a value is NA.
- Many R functions have an argument na.rm=. Setting it to TRUE allows the function to automatically drop NAs.
  > is.na(x) # Shows which element is missing
  [1] FALSE FALSE TRUE
  > mean(x) # Returns NA
  [1] NA
  > mean(x, na.rm=T) # Omits missing values
  [1] 0.5
- Other special values similar to NA include NaN ("not a number") and Inf.
  > 1/0
  [1] Inf
  > 0/0
  [1] NaN
  > is.na(c(1/0, 0/0, NA)) # NaN is treated as NA; Inf is not
  [1] FALSE TRUE TRUE
2.6 Exporting Data Created in R

- Once finished with manipulating a data frame, you might want to save it into a non-R file.

- The `write.table(x, file)` function saves the object `x` as an ASCII file, with its name specified by the `file` argument.

- Alternatively, you can save it into a csv file by `write.csv(x, file)`

```r
> write.table(Africa.sub, file="africa_sub.txt")
> write.csv(Africa.sub, file="africa_sub.csv")
```