Supplementary Information for correspondence by:
Mark P. Styczynski*[1], Kyle L. Jensen*[1], Isidore Rigoutsos[2], Gregory N. Stephanopoulos[1]
[1] Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[2] IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
* These authors contributed equally to this work

Working title: "Miscalculations in BLOSUM62 surprisingly improve search performance"

Supplementary Methods

Matrices

Matrices were created using the methods described previously[1]. The programs and Blocks training data used to create the original BLOSUM series of matrices were obtained from ftp://ftp.ncbi.nih.gov/repository/blocks/unix/blosum/blosum.tar.Z. Additional, updated programs were obtained from ftp://ftp.ncbi.nih.gov/repository/blocks/unix/blosum/programs and ftp://ftp.ncbi.nih.gov/repository/blocks/unix/blimps. All matrices were created using the same Blocks 5 release available when the BLOSUM matrices were initially published.

Databases

We used the ASTRAL[2] database as the database for our performance-evaluating searches. Our method was designed to emulate the work by Price et al.[3]. ASTRAL is created based on the SCOP database[4], which classifies proteins based on their function, structure, and sequence into a hierarchical structure of classes, folds, superfamilies, and families. Sequences in the same superfamily can have low sequence similarity, but are likely to have a common evolutionary origin based on their structural and functional features. Because these

classifications are made by human inspection, not via automated sequence alignment procedures, it makes an ideal "gold standard" for remote homolog detection tests.

From the full set of ASTRAL genetic domain sequences, we chose the sequence set from which 40% identical sequences had been eliminated. By using this subset, our search focuses on the detection of remote homologs that are more challenging for substitution matrices to discover and thus will differentiate the abilities of the respective matrices to find distant relatives. The sequences were further filtered by pseg[5] for the removal of low-complexity regions. The unfiltered sequence set is available on-line from the ASTRAL database at http://astral.berkeley.edu/scopseq-1.69/astral-scopdom-seqres-gd-sel-gs-bib-40-1.69.fa. This non-redundant set numbers 7290 sequences. Each sequence was extracted from the database one at a time and used as a query for the entire database. Search results in the same superfamily as the query were considered to be true positives.

Search methods

We used both the Smith-Waterman[6] local alignment algorithm and BLAST[7] for searches against the databases. In particular, we used the ssearch implementation of the Smith-Waterman algorithm by Pearson[8,9] and the NCBI version of BLAST in the C Toolbox, obtained from ftp://ftp.ncbi.nlm.nih.gov/toolbox/ncbi_tools/, which gives the same results as the commonly-used web interface at NCBI. While Smith-Waterman is an exhaustive, sensitive search that may give a more accurate assessment of a given matrix's performance capability, BLAST is a faster search that is almost an "industry standard" and so better represents the impact of different matrices on the "average user". In addition, BLAST can be run in an ungapped fashion to eliminate the effects of gap penalty parameters.

For our Smith-Waterman database searches, we mostly used the default parameters of the ssearch program, with one notable exception: we varied the gap initiation (existence) penalty from -6 to -14 and the gap extension penalty from -1 to -2 for the initial and revised BLOSUM matrices we investigated. (In general, we expect relatively broad maxima of approximately optimal penalty values, as seen previously[10]). For matrices with different scales, we changed the gap parameters accordingly.

For our BLAST searches, we varied the gap penalty parameters as in the ssearch sections. Appropriate lambda, K, alpha, and beta parameters for significance estimation were calculated using the island method described elsewhere[11,12]. Calculations were performed using routines provided by Stephen Altschul[13]. Other than gap penalties, all other default parameters for the toolbox version of BLAST version 2.2.13 (12/6/2005) "blastall" were used. Notably, this does not include composition-based adjustments of E-values, which are now used by default on the NCBI web interface of BLAST. Previously[14], these adjustments have been shown to actually decrease performance in individual protein database BLAST searches.

Performance evaluation

We used the Bayesian bootstrap method presented elsewhere[3] to evaluate the statistical significance of the mean difference in coverage between any two substitution matrices in our ASTRAL-based tests. This method uses coverage vs. errors per query as a means to evaluate the effectiveness of different substitution matrices, where coverage is defined simply as the fraction of true positives found at a given errors per query threshold. Concerted Bayesian bootstrapping is used to determine the statistical significance of the difference in matrices' effectiveness by evaluating whether slightly different reference databases would have yielded different coverage

vs. errors per query curves. True positives were identified as described above. The best-performing gap parameters for each respective matrix were used to compare performance.

It is worth noting that the most appropriate way to compare two families of matrices is via entropy analogues. A matrix's relative entropy is a reflection of the required minimum length of homology necessary for a potential "match" or "hit" to be distinguishable from noise[15]. Comparing matrices with different entropies would be an inappropriate comparison as the matrices could be tuned to find homologous regions of different lengths. By comparing matrices with the same relative entropy, we can better assess the "value" or "correctness" of the information encoded in the matrices. As such, all matrices in this work that were used to search any database were (unless otherwise noted) chosen to have approximately the same relative entropy as BLOSUM62, which was 0.6979. Previous work[16] has further shown that matrices with entropy values of about 0.7 typically have the best average performance within a substitution matrix family.

Availability

This supplementary information, along with the original and revised blosum.c programs, are available at http://web.mit.edu/bamel/blosum.

An overview of the clustering and normalization process

One of the errors in the original BLOSUM program is an incorrect normalization of calculated values in the re-clustering step. We will now briefly describe the purpose and workings of the re-clustering step along with the error that we identified.

When two sequences are compared to find their substitution frequencies, weighting consistent with the described algorithm would require that the contribution of such a pair be divided (normalized) by the size of the two clusters to which the sequences have been assigned, so that the overall contribution of each cluster is equal to just "one protein". However, as implemented in the program, the weight is normalized by only one cluster size, not both. As an example, we analyze this vastly simplified block consisting of six sequences of length 3:

```
KLW
KLA
RLW

KKL
VKL

PIL
```

If we were to re-cluster these sequences at 62% identity, then they would be in the three groups as indicated by the spacing above: `KLW` is 67% similar to both `KLA` and `RLW`, so those three form one cluster even though `KLA` and `RLW` are only 33% similar. For the purposes of determining the number of "substitutions" at any given position, each cluster is essentially considered as "one protein". In this example, then, there are only three "proteins" after clustering, meaning there should only be three total substitution pairs per position (substitutions between "proteins" formed by clusters 1 & 2, 1 & 3, and 2 & 3). However, we should not throw away the information encoded by all of the possible substitutions from the members of the clusters, so we will count substitutions between all sequences that are not in the same cluster and weight them appropriately such that we only have a total of three substitution pairs. This serves to decrease the influence of highly similar sequences that have not diverged much. To perform this weighting, we should normalize each substitution, or pair, by the product of the two cluster sizes. We will use the first column as an example: for the K-K pair within the first cluster, there

would be no contribution (since all of the sequences in the cluster are essentially "one" protein). For the K-K pairs formed between the first and second clusters, there would be 2 pairs of weight 1/6, resulting in a total of 1/3 K-K pairs. Proceeding through the rest of the pairs, the total weights would be 1/6 for R-K, 1/3 for K-V, 7/6 for K-P, 1/6 for R-V, 1/3 for R-P, and 1/2 for V-P, which adds up to a total of three pairs, as expected.

However, in the original blosum.c program, only the size of the "second" cluster in a pair was used for normalization. The loop in the program proceeded from the first sequence encountered, so the only normalization would be the size of the cluster of the later sequence. For the case above, the K-K pairs would each be normalized by 1/2 instead of 1/(2 x 3), leaving 1 K-K pair. The rest of the weights are 1/2 for R-K, 1 for K-V, 3 for K-P, 1/2 for R-V, 1 for R-P, and 1 for V-P, giving a total of 8 pairs. Not only does the absolute number of pairs change from what should be computed, but the relative amounts of pairs change as well. This inconsistency has since been fixed in updated versions of blosum.c as a part of another package[17], though we were unable to find any published analysis of its impact on the BLOSUM matrices or any other programs that use the revised matrices. That is, all of the commonly-used search tools that we surveyed use the original, and not revised and "correct", version of BLOSUM62.

Supplementary Discussion

Order-dependent differences in BLOSUM62 are statistically significant

The normalization error causes the BLOSUM matrices to be order-dependent, meaning that the BLOSUM matrix currently in widespread use is not the only possible "equivalent" matrix that we could be using. A simple demonstration of this can be seen by merely rearranging the order of the sequences in the input file (while still conserving the sequences in any given

block). We shuffled the order of the sequences within each block, each time obtaining different "versions" of BLOSUM62. (Our revised blosum.c code was insensitive to ordering within blocks in identical tests.)

We then analyzed two shuffled versions with both ssearch and BLAST, and analyzed another five only with BLAST. Each of these shuffled versions of the Blocks database needed a BLOSUM re-clustering at 63% to make a matrix isentropic with BLOSUM62 (see Supplementary Methods). Performance was measured by "coverage", or the fraction of true positive homologies uncovered at a given errors per query threshold for a hand-curated gold standard dataset (see Supplementary Methods). These "shuffled" matrices frequently showed performances different (to a statistically significant degree as assessed via Bayesian bootstrapping; see Supplementary Methods) from the original BLOSUM62 matrix. Supplementary Figure 1 is a summary of these differences plotted on the same axis; Supplementary Figure 2 plots each matrix's performance individually. Over the expected optima of gapped BLAST searches, no matrices were consistently better than the original BLOSUM62, and about half of them were consistently worse. In Supplementary Figure 1, a representative "average" matrix's performance is plotted along with the "best" matrix's performance and the "worst" matrix's performance. In Supplementary Figure 2, we see that versions 1 and 2 of the shuffled BLOSUM matrix perform similarly in both BLAST and ssearch tests; thus, one may expect BLAST performance to be a reasonable (qualitative) predictor of ssearch performance for the other five versions.

We also found that the entropy of the current BLOSUM62 matrix was aberrantly high, with a p-value of 1.6×10^{-5} based on 425,400 matrices generated at a re-clustering percentage of 62% from shuffled Blocks databases. No simple cluster sorting method

(e.g., size of clusters) could create similarly aberrant entropies. Despite this statistically significant deviation in matrix properties, we were unable to correlate high entropy at constant re-clustering percentage with increased matrix performance (data not shown).

BLOSUM62 performs better than the "correct" RBLOSUM64 matrix

Perhaps the most interesting result of our work is that, as noted in the main text, if the matrices had been initially created as they were intended, they would have been less effective than they currently are, as illustrated in Figure 1. That is, the RBLOSUM64 matrix that is isentropic to BLOSUM62 performs consistently worse than BLOSUM62. The difference in performance between the two matrices is statistically significant across a wide range of EPQ values and in a wide range of search settings. For Smith-Waterman searches, this difference is seen at almost all combinations of gap penalty values (data not shown). Only for penalty values with extremely poor performances is this trend not observed, and even in these few cases RBLOSUM64 never performs statistically significantly better than BLOSUM62. In gapped BLAST searches, the same trend is observed (data not shown). BLOSUM62 performs better than RBLOSUM64 in ungapped BLAST searches as well (data not shown). It is also worth noting that RBLOSUM64 performs better than RBLOSUM62 in gapped BLAST searches (data not shown), supporting the previously stated hypothesis regarding matrices with relative entropy of 0.7.

Variation introduced by errors is greater than cross-validation variation

To further analyze the difference between the performances of the BLOSUM62 and RBLOSUM64 matrices, we looked at the "distributions" of these types of matrices. We used the

isentropic shuffled BLOSUM matrices as a sample distribution to represent the matrices that could have been formed with only a different database ordering. We used 10–fold cross–validation on the Blocks database to infer the inherent uncertainty or error in the RBLOSUM64 matrix. Even under the relatively harsh method of cross-validation for RBLOSUM64 (compared to merely changing the order, not content, of the database for BLOSUM62), we see that there are some striking differences between the two distributions (see Supplementary Figure 3). First, it seems that the distribution of performances for matrices based on shuffled Blocks databases is quite a bit larger than the distribution of performances for the cross-validation RBLOSUM matrices. In addition, it appears that BLOSUM62 is not necessarily representative of the "mean" matrix that could have been obtained using the original blosum.c code. That is, it performs as good as any of a variety of variants from shuffled Blocks databases, and better than most. Clearly, there is something (at least qualitatively) different between the two implementations, and the deviation of BLOSUM62 from RBLOSUM64 is more than one would expect from the error and uncertainty inherent to RBLOSUM64, the matrix that "should" have been created.

Increasing matrix scale provides mixed results

In another set of experiments, we increased the scale of the matrices used in ssearch queries. BLOSUM62 values are scaled in half-bits, meaning that the base-2 log-likelihood raw scores computed from the training set substitution frequencies are multiplied by 2 before being rounded off to integers for ease of use. If, instead, we multiply those raw scores by 3 before rounding them, we would expect that the performance of the matrix would at least stay the same, if not improve from truncating less information content in the matrix values. While this was true for the RBLOSUM64 matrix, which had small (though not statistically significant)

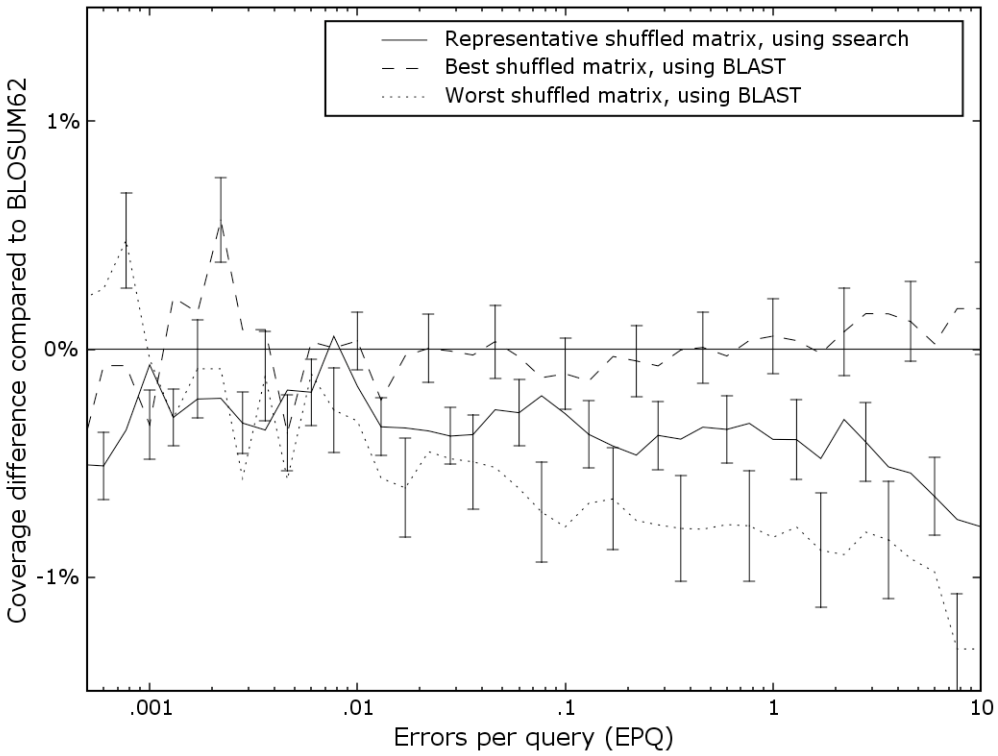
improvements, BLOSUM62 actually had statistically significant decreases in performance (using ssearch; data not shown). These two third-bit matrices performed very similarly, indicating that perhaps the intrinsic difference between them is not very great. In fact, the RMSD for the raw values in the matrices is only 0.104, supporting the idea that they are not too intrinsically different before scaling and rounding.

BLOSUM62 performance is a complex effect of random factors

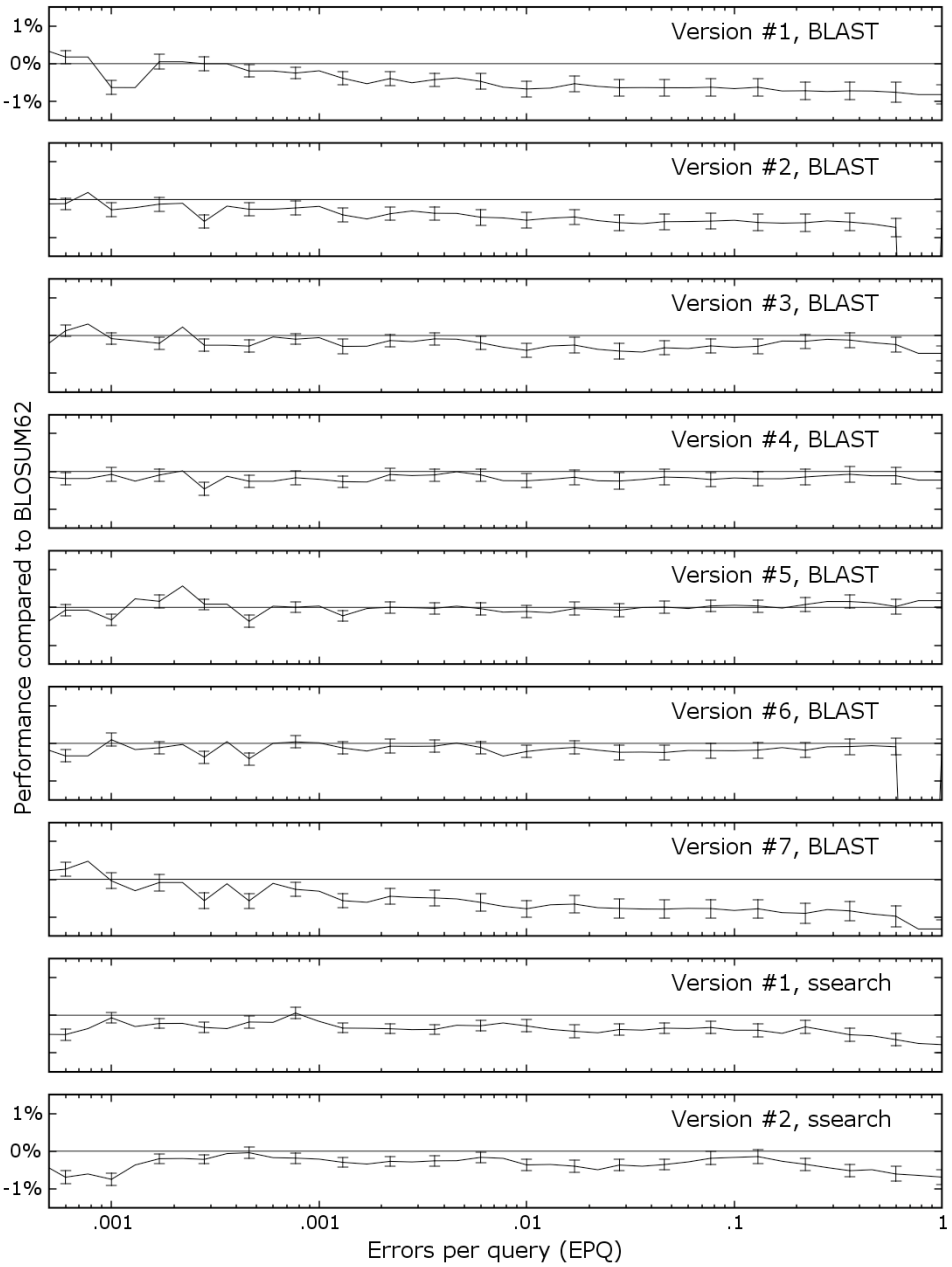
As discussed above, the order of the sequences in the Blocks database had a profound effect on the values in the BLOSUM62 matrix and the effectiveness of that matrix. It seems that, fortuitously, one of the best possible arrangements was used. Interestingly enough, this arrangement is partly a simple matter of alphabetization: first all clusters with only one sequence are listed in alphabetical order, and then all clusters with multiple sequences are listed in an order such that the first sequences across all of these clusters are in alphabetical order. With any different implementation, the effectiveness of BLOSUM62 may have been acceptable, but quite different.

We believe that the slight performance boost in BLOSUM62 relative to RBLOSUM64 is largely due to artifacts of truncation error. While BLOSUM62 has abnormally high entropy for a re-clustering value of 62% with respect to other matrices that could have been formed with the original BLOSUM implementation, this higher entropy does not correlate with performance. It has been noted that the entropy of the BLOSUM62 matrix based on its scaled and rounded values deviates noticeably more from the raw matrix's entropy than one typically sees for other scaled and rounded matrices[18]. While using these scaled and rounded entropies to attempt "isentropic" comparisons partially resolves the ssearch performance difference between

BLOSUM62 and RBLOSUM64, it only exaggerates the problem in BLAST searches (data not shown). While this entropy-related anomaly does not completely explain the performance difference of BLOSUM62, it does point out that there may be something different about the scaling and rounding of BLOSUM62 relative to other scoring matrices. Combining this observation with the fact that a larger scale for BLOSUM62 actually leads to a decrease in performance, we can then reasonably infer that the improved performance of BLOSUM62 is due to nothing more than a rare set of circumstances caused by miscalculated normalizations, low-resolution scaling, and some fortuitous rounding. This conclusion is actually quite fascinating, as it indicates that there is still more to learn about amino acid substitution frequencies: if essentially random processes (rounding and normalization errors) can cause a statistically significant increase in performance for our best rational estimates of these frequencies, then there is likely some fundamental aspect of modeling amino acid substitution that we have yet to grasp or capture.

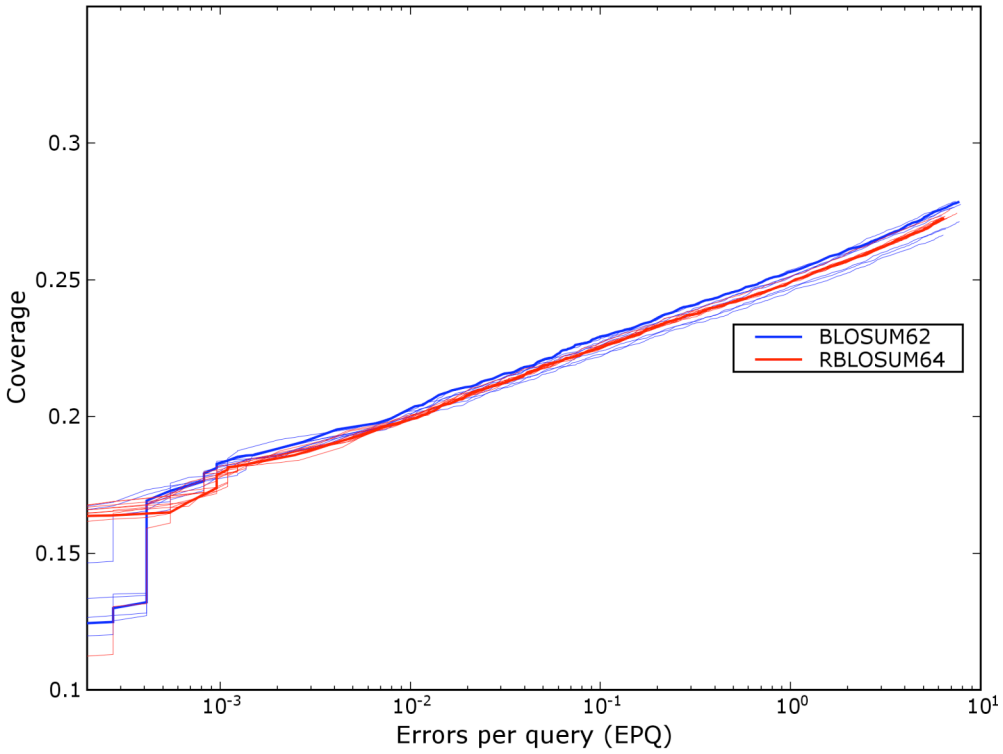


Supplementary Figure 1. Performance difference between individual BLOSUM matrices made from shuffled databases and the widely used BLOSUM62. "Coverage" is the fraction of true positive homologies found at a given errors per query (EPQ) threshold for a hand–curated gold standard dataset (see Supplementary Methods). Negative values indicate that the original BLOSUM62 performs better. Error bars indicate 95% confidence intervals using Bayesian bootstrapping methods. Both shuffled versions analyzed with ssearch performed similarly (solid line). Of the seven versions analyzed with BLAST, two performed slightly worse than the best matrix (dashed line) and were mostly indistinguishable from BLOSUM62. Three matrices performed similarly to the worst matrix (dotted line), showing statistically significant differences in performance across most EPQ values.



Supplementary Figure 2. BLAST and ssearch coverage performance difference between BLOSUM62 and matrices formed from shuffled versions of the Blocks database. The results

from each matrix's best–performing gap penalty values were used. Negative values indicate that the original BLOSUM62 is better. Error bars indicate 95% confidence intervals using Bayesian bootstrapping methods. Error bars that do not cross the origin indicate statistically significant differences between BLOSUM62 and the matrix being plotted. Gap extension penalties for all matrices were 1. Existence penalties were 9 for both matrices when using ssearch and 11 for shuffled versions 2, 4, 5, and 6 when using BLAST; shuffled versions 1, 3, and 7 used existence penalties of 10.



Supplementary Figure 3. A coverage vs. errors-per-query (CVE) plot for BLOSUM62 and RBLOSUM64 performance distributions using gapped BLAST. Rather than Bayesian bootstraps, we display versions of the BLOSUM62 matrix derived from Blocks databases with shuffled blocks and versions of the RBLOSUM64 matrix derived from Blocks databases each with 10% of the blocks removed (10-fold cross-validation). These RBLOSUM64 matrices help to indicate the training set–dependent error that is inherent to a matrix rigorously derived from Blocks. We note that the variation of the RBLOSUM64 matrices is less than that of the BLOSUM62 matrices, and that the original BLOSUM62 matrix (the darker line) is not representative of the "mean" performance one could have expected. BLOSUM62 (and shuffled derivatives) gap penalties are as described in Supplementary Figure 1, while all RBLOSUM64 reduced–training set (existence, extension) gap penalties were (10,1) except for one matrix,

which used (11,1).

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X
A	4	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	-1
R	-2	5	0	-2	-3	1	0	-2	0	-3	-2	2	-2	-3	-2	-1	-1	-2	-2	-3	-1	0	-1
N	-1	0	6	1	-3	0	0	-1	1	-3	-3	0	-2	-3	-2	0	0	-3	-2	-3	3	0	-1
D	-2	-2	1	6	-3	0	2	-2	-1	-3	-3	-1	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1
C	-1	-3	-3	-3	9	-3	-4	-3	-2	-1	-1	-3	-1	-2	-3	-1	-1	-3	-2	-1	-3	-3	-2
Q	-1	1	0	0	-3	5	2	-2	1	-3	-2	1	0	-3	-1	0	0	-2	-2	-2	0	3	-1
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1
G	0	-2	-1	-2	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	-1	-2	-2	-3	-3	-1	-2	-2
H	-2	0	1	-1	-2	1	0	-2	7	-3	-3	-1	-1	-1	-2	-1	-2	-1	1	-3	0	0	-1
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-2	-1	2	-3	-3	-1
L	-2	-2	-3	-3	-1	-2	-3	-4	-3	2	4	-2	2	1	-3	-2	-1	-1	-1	1	-3	-2	-1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1
M	-1	-2	-2	-3	-1	0	-2	-3	-1	1	2	-1	6	0	-2	-1	-1	-2	-1	0	-2	-1	-1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	1	-3	0	6	-3	-2	-2	1	3	-1	-3	-3	-1
P	-1	-2	-2	-2	-3	-1	-1	-2	-2	-3	-3	-1	-2	-3	7	-1	-1	-3	-3	-2	-2	-1	-2
S	1	-1	0	0	-1	0	0	-1	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	-1
T	0	-1	0	-1	-1	0	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-3	-2	0	-1	-1	-1
W	-3	-2	-3	-4	-3	-2	-3	-2	-1	-2	-1	-3	-2	1	-3	-3	-3	11	2	-3	-3	-3	-2
Y	-2	-2	-2	-3	-2	-2	-2	-3	1	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-2	-2	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	2	1	-2	0	-1	-2	-2	0	-3	-1	4	-3	-2	-1
B	-2	-1	3	4	-3	0	1	-1	0	-3	-3	0	-2	-3	-2	0	-1	-3	-2	-3	4	0	-1
Z	-1	0	0	1	-3	3	4	-2	0	-3	-2	1	-1	-3	-1	0	-1	-3	-2	-2	0	3	-1
X	-1	-1	-1	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1

Supplementary Figure 4. The revised BLOSUM matrix, RBLOSUM64. Values in red are one greater than they would be in BLOSUM62, while values in green are one less than they would be in BLOSUM62. These matrices differ in roughly 15% of their values.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X
A	4	-1	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	-1
R	-1	5	0	-1	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-2	-1	-2	-1	0	-1
N	-1	0	5	1	-2	0	0	-1	1	-3	-3	0	-2	-2	-2	0	0	-3	-2	-3	3	0	-1
D	-2	-1	1	6	-3	0	2	-1	-1	-3	-3	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1
C	-1	-3	-2	-3	9	-3	-3	-3	-2	-1	-1	-3	-1	-2	-3	-1	-1	-3	-2	-1	-3	-3	-2
Q	-1	1	0	0	-3	5	2	-2	1	-2	-2	1	0	-3	-1	0	0	-2	-1	-2	0	3	-1
E	-1	0	0	2	-3	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1
G	0	-2	-1	-1	-3	-2	-2	5	-2	-4	-4	-2	-3	-3	-2	-1	-2	-3	-3	-3	-1	-2	-1
H	-2	0	1	-1	-2	1	0	-2	7	-3	-2	-1	-1	-1	-2	-1	-2	-1	1	-3	0	0	-1
I	-1	-3	-3	-3	-1	-2	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-2	-1	2	-3	-3	-1
L	-2	-2	-3	-3	-1	-2	-3	-4	-2	2	4	-2	2	1	-3	-2	-1	-1	-1	1	-3	-2	-1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	4	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1
M	-1	-1	-2	-3	-1	0	-2	-3	-1	1	2	-1	6	0	-2	-1	-1	-2	-1	0	-2	-1	-1
F	-2	-3	-2	-3	-2	-3	-3	-3	-1	0	1	-3	0	6	-3	-2	-2	1	3	-1	-3	-3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-3	7	-1	-1	-3	-3	-2	-1	-1	-1
S	1	-1	0	0	-1	0	0	-1	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	-1
T	0	-1	0	-1	-1	0	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-3	-2	0	-1	-1	-1
W	-3	-2	-3	-4	-3	-2	-3	-3	-1	-2	-1	-3	-2	1	-3	-3	-3	11	2	-2	-3	-3	-2
Y	-2	-1	-2	-3	-2	-1	-2	-3	1	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-2	-2	-1
V	0	-2	-3	-3	-1	-2	-2	-3	-3	2	1	-2	0	-1	-2	-2	0	-2	-1	4	-3	-2	-1
B	-2	-1	3	4	-3	0	1	-1	0	-3	-3	0	-2	-3	-1	0	-1	-3	-2	-3	3	0	-1
Z	-1	0	0	1	-3	3	4	-2	0	-3	-2	1	-1	-3	-1	0	-1	-3	-2	-2	0	3	-1
X	-1	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	-1	-1	-1	-1	-1

Supplementary Figure 5. The revised BLOSUM matrix, RBLOSUM62. This matrix is included here only for reference; it was not used for any analyses plotted in this paper, because it is not isentropic to the original BLOSUM62 matrix. (We also found that its BLAST performance was inferior to RBLOSUM64.) Again, values in red are one greater than they would be in BLOSUM62, while values in green are one less than they would be in BLOSUM62.

---

[1] Henikoff, S and Henikoff, JG. *Proc Natl Acad Sci U S A* **89**, 10915-10919 (1992).
[2] Brenner, SE, Koehl, P, and Levitt, M. *Nucleic Acids Res* **28**, 254-256 (2000).
[3] Price, GA, Crooks, GE, Green, RE, and Brenner, SE. *Bioinformatics* **21**, 3824-3831 (2005).
[4] Murzin, AG, Brenner, SE, Hubbard, T, and Chothia, C. *J Mol Biol* **247**, 536-540 (1995).
[5] Wooton, JC and Federhen, S. *Comput Chem* **17**, 149-163 (1993).
[6] Smith, TF and Waterman, MS. *J Mol Biol* **147**, 195-197 (1981).

[7] Altschul, SF, Gish, W, Miller, W, Myers, EW, and Lipman, DJ. *J Mol Biol* **215**, 403-410 (1990).
[8] Pearson, WR. *Methods Enzymol* **183**, 63-98 (1990).
[9] Pearson, WR. *Genomics* **11**, 635-650 (1991).
[10] Gribskov, M and Robinson, NL. *Comput Chem* **20**, 25-33 (1996).
[11] Olsen, R, Bundschuh, R, and Hwa, T. *Proc Int Conf Intell Syst Mol Biol*, 211-222 (1999).
[12] Altschul, SF, Bundschuh, R, Olsen, R, and Hwa, T. *Nucleic Acids Res* **29**, 351-361 (2001).
[13] Altschul, SF. National Center for Biotechnology Information. Personal communication (2005).
[14] Schaffer, AA, et al. *Nucleic Acids Res* **29**, 2994-3005 (2001).
[15] Altschul, SF. *J Mol Biol* **219**, 555-565 (1991).
[16] Henikoff, S and Henikoff, JG. *Proteins* **17**, 49-61 (1993).
[17] Henikoff, S, Henikoff, JG, Alford, WJ, and Pietrokovski, S. *Gene* **162**, GC17-26 (1995).
[18] Altschul, SF. National Center for Biotechnology Information. Personal communication (2006).