Using Blocks in Pairwise Sequence Alignment

Joe Meehan
December 6, 2002

Biochemistry 218
Computational Molecular Biology

Introduction

Since the introduction of dynamic programming techniques in the early 1970's several related algorithms have been developed that make it relatively easy to align two biological sequences, most notably Needleman-Wunsch for global alignments and Smith-Waterman for local alignments. However, even with these time-tested algorithms, not all carefully generated alignments are good alignments. It is not hard to generate an alignment that looks good on paper, but fails to correctly match known sequence similarities, or which puts gaps in places that almost certainly should not have gaps. The essential problem is that the alignment programs incorporate no external knowledge of the sequences being aligned. They have no way of ensuring that similar structures align, because the known structural information is unavailable to them. The task of painstakingly evaluating proposed alignments in the light of known biological information is left to the researcher.

This paper and associated software explore one way in which external knowledge could be incorporated into pairwise sequence alignment software. A prototype sequence alignment program is presented which allows the researcher to incorporate information from databases such as BLOCKS+ or eBlocks into the sequence alignment process, either as checks against the proposed alignment or as hard constraints on the alignment. Some example alignments generated by this method are shown, and compared to alignments generated by similar methods.

The patterns stored in the various block databases do not necessarily represent the same structural details that a biologist might check in an alignment, but they do represent conserved subsequences that the biologist would also expect to match in a valid sequence alignment. Incorporating the knowledge represented in the block databases into the sequence alignment process might help the biologist generate alignments that are consistent with current knowledge of protein families.

Program Overview

The program presented here is meant as a prototype implementation for exploring the inclusion of block information in pairwise sequence alignment. It allows the user to enter the location of up to five blocks in the two sequences, and uses that information to either identify the block locations in traditional alignments, or force the alignment of the identified blocks. Identifying block locations is not difficult to code, but it makes it much easier for the researcher to identify block mismatches. Forcing the alignment of blocks shows the researcher how the sequences could be aligned given the constraint that like blocks must align. The program currently uses a Needleman-Wunsch alignment with

affine gap scores (Durbin *et al*, 1998), but in theory it is not tied to any particular alignment algorithm.

Ideally, a pairwise sequence alignment program that used block information would access a blocks database and find the common blocks in the two sequences automatically, saving the researcher the time and trouble of doing the searches separately. Due to time limitations the current prototype does not do the block database searches, but if the prototype proves useful this would be a worthwhile enhancement.

Similar Approaches

This is not the first attempt to meld motifs and sequence alignments. Two previous efforts are particularly noteworthy: PHI-BLAST and SWP.

PHI-BLAST (Altschul *et al*, 1997) is more of a motif driven search engine than a simple pairwise alignment tool. Given a sequence and a PROSITE-format regular expression describing a pattern within the sequence, it will search a database for other sequences that include the pattern and have significant similarity to the sequence. It can then display pairwise alignments of the found sequences with the initial sequence, aligned around the supplied pattern. The alignments are based on the BLAST algorithm, and in fact use much of the same code.

PHI-BLAST differs from the program presented here in its focus on database searches, its use of regular expressions rather than blocks, and its use of the BLAST local alignment. However, in its combined use of motifs and sequence alignment it is close in spirit to the current effort. The results of the current program on an example alignment will be compared to PHI-BLAST later in this paper.

A paper published earlier this year (Comet and Henry, 2002), presents an algorithm designed specifically to incorporate patterns into pairwise sequence alignment. The algorithm, named Smith & Waterman algorithm with Patterns (SWP), modifies the Smith-Waterman recurrence relation so that alignment choices that result in pattern matches are given additional weight. Like PHI-BLAST, SWP uses PROSITE-derived regular expressions as patterns. Unlike PHI-BLAST, SWP does not force pattern matches, but strongly encourages them through weighting.

The SWP paper makes a number of interesting points. First, it goes to great lengths to show that the classic Smith-Waterman algorithm does not always align common patterns in two sequences. After making pairwise Smith-Waterman alignments of all proteins in all pattern-defined protein families in PROSITE, Comet and Henry conclude that in

general Smith-Waterman correctly aligns patterns when the patterns occur in the regions of highest similarity between the two sequences. But, two sequences sometimes share patterns that do not occur in the regions of highest similarity, and in these cases Smith-Waterman does not always match the patterns. This provides motivation for a pairwise alignment program that knows about the patterns and will try to match them.

Another interesting point from the SWP paper is that sometimes two sequences have patterns in common, but the patterns do not occur in the same order in both sequences due to inversion. In this case, the biologist's natural tendency to force the alignment of both patterns cannot succeed. SWP cannot match both patterns either, so it chooses which pattern to align according to the strength of the matches between the two patterns.

A third point, and a complicating factor for the alignment algorithm, is that matches to the same PROSITE regular expression in two sequences can have different lengths, because regular expressions can contain variable length subpatterns such as x[6,8]. Much work on SWP was devoted to developing appropriate algorithms for aligning variable length patterns.

Program Design

The current program allows a researcher to enter two protein sequences and identify the locations of up to five common blocks within the two sequences. It allows the researcher to specify a gap penalty and a gap extension penalty, and choose from a small set of substitution matrices. A check box is used to select one of two program behaviors, either force the specified blocks to align, or simply show the block locations in the alignment to see if they match and where mismatches occur.

The program performs a Needleman-Wunsch global alignment with affine gap scores as described by Durbin, however other global alignments could be easily supported. Local alignments could also be supported with modifications to the routines that show block locations, as these routines currently assume that the entire sequence appears in the alignment.

Unlike SWP, pattern information in this program is added to sequence alignment without any modification to the basic alignment algorithms. In fact, the basic alignment routines are taken without modification from Peter Sestoft's match2 code (Sestoft, 1999) based on the Durbin book. Sestoft's code was enhanced in the following manner:

- The user interface was completely rewritten to allow entry of block information, gap penalty, gap extension penalty, substitution matrix, and the option to force alignments.

- A new substitution matrix, Blosum30, was implemented in addition to the provided Blosum50 matrix.

- The routine that displays alignments was greatly enhanced to include display of block locations, to mark identities with a bar character ("|"), to insert line breaks after 50 characters, and to display the location within a sequence at the end of each line.

- New top-level logic was added above the alignment routines to implement the forced alignment of blocks as described below.

If the researcher does not choose to force block alignments, the program performs a normal alignment, but adds symbols to the alignment display to indicate where the blocks are located within each sequence. Each block is labeled by a block number, from 1 through 5, in the alignment display.

This is a simple addition to the alignment display, but it makes it much easier for the researcher to determine if a normal alignment will align the blocks, and whether it has placed any gaps within blocks. Also, it makes it easy for the researcher to experiment with different gap penalties and gap extension penalties to see if changes there will affect the block alignments. (They quite often do.)

If the blocks do not align, the researcher can choose to force block alignment. In this case, the program partitions the sequences into subsequences defined by the blocks, and aligns each partition separately. For instance, if two sequences share one block, the program will first perform an alignment of the subsequence before the block, then it will display the matched block, and finally it will align the subsequence after the block. By extension, if there is more than one block, the program will align subsequences between corresponding blocks until there are no more blocks, and finally it will align the ends of the sequences.

Again, this is a somewhat simplistic approach. It merely automates what Comet and Henry call, "the method used by biologists which consists in forcing the alignment of occurrences of motifs." They note that forcing alignments in this way does not handle cases of inversion where two blocks appear in both sequences, but in a different order. In such a situation, SWP cannot align both blocks, but it does automate the choice between the two. It is not clear how often this situation occurs; however, the current program

could be modified to detect the disordered blocks and either refuse to do the alignment or choose which block to use based on the strength of the block similarity.

Using the Program

The program is accessible via the worldwide web at http://www.stanford.edu/~jmeehan/AlignBlocks.html. It is implemented as a Java applet, and requires a Java virtual machine to run. Most modern browsers support Java natively or support a plug-in from Sun (http://java.sun.com/getjava/download.html). The applet architecture was chosen for this prototype because it has few dependencies on the browser and web server, a desirable feature when the programmer controls neither the browser nor the web server.

There are some unfortunate limitations of the applet architecture that make the prototype awkward at times. Most notably, paste and copy functions are only supported by the keyboard equivalents, Control-V and Control-C on Windows machines, or Command-V and Command-C on Macintosh, rather than as normal GUI menu options. This limitation can be overcome by signing the applet, but since signing is browser-dependent it was not implemented in the prototype. Also, because the Java byte code must be downloaded and run locally, the program is can slow, especially on slow machines.

A production version of the program would be better implemented as a Java servlet or Java Server Page (JSP). Either option would require support of the web server.

An Example

The Cystatin family of cysteine protease inhibitors, PROSITE entry PS00287, is a family of over 50 protein sequences that have previously been shown to be evolutionarily related (Rawlings *et al*, 1990). Since the family was originally defined several new members have been discovered (Cornwall *et al*, 1999). This sort of situation should be a good application for block-aware pairwise sequence alignment, because there are new members to a family with well-defined blocks.

Here we compare the sequences of CYTC_HUMAN, an original family member used in block definitions, with CST8_HUMAN, a newly discovered family member that does not appear in the block definitions. In order to facilitate comparison between the current program and PHI-BLAST, the eBlocks program (Su *et al*, 2002) was used to search both for common blocks and common regular expression motifs. One common block was found (P22085G3B1), as well as one common motif (LxFx[ILMV]xExN[KR]x[ST]xD). The block locations are as follows:

	CYTC_HUMAN		CST8_HUMAN	
Block Name	Start – End	E-Value	Start - End	E-Value
P22085G3B1	41 - 66	4.56e-14	37 - 62	8.75e-02

After entering the block location for both sequences into the program, an initial alignment was performed. A Blosum30 matrix was used because Cornwall reports a 27% sequence similarity (Cornwall *et al*, 1999). As can be seen below, the program shows that the alignment introduces a gap in CST8_HUMAN that causes the blocks to be misaligned. (The top sequence is CYTC_HUMAN.)

```
Gap Penalty 16 Gap Extension 4 Blosum30 substitution matrix

AFFINE GLOBAL:
Score = 280
                                          1111111111
MAGPLRAPLLLLAILAVALAVSPAAGSSPGKPPRLVGGPMDASVEEEGVR 50
|       | || |     |          |    |   |  ||     |
MPRCRWLSLILLTIPLALVARKDPKKNETGVLRKL--KPVNAS--NANVK 46
                                       11111  11111
1111111111111111
RALDFAVGEYNKASNDMYHSRALQVVRARKQIVAGVNYFLDVELGRTTCT 100
  | ||  |||| | | |          |  |      |  |||  |  |
QCLWFAMQEYNKESEDKYVFLVVKTLQAQLQVTNLLEYLIDVEIARSDCR 96
1111111111111111

KTQPNLDNCPFHDQPHLKRKAFCSFQIYAVPWQGTMTLSKSTCQDA     146
|       |       ||||  |||   | || |  |     | ||
KPLSTNEICAIQENSKLKRKLSCSFLVGALPWNGEFTVMEKKCEDA     142
```

In this case, it is possible to modify the alignment parameters to bring the blocks into alignment without using the force blocks option. For instance, simulating the original Needleman-Wunsch algorithm by raising the gap extension penalty to match the gap penalty of 16 has this effect:

```
AFFINE GLOBAL:
Score = 259
                                          1111111111
MAGPLRAPLLLLAILAVALAVSPAAGSSPGKPPRLVGGPMDASVEEEGVR 50
|  | |   | |  |   ||               |             |
M--P-RCRWLSLILLTIPLALVARKDPKKNETGVLRKL-KPVNASNANVK 46
                                          1111111111

1111111111111111
RALDFAVGEYNKASNDMYHSRALQVVRARKQIVAGVNYFLDVELGRTTCT 100
  | ||  |||| | | |          |  |      |  |||  |  |
QCLWFAMQEYNKESEDKYVFLVVKTLQAQLQVTNLLEYLIDVEIARSDCR 96
1111111111111111
```

```
KTQPNLDNCPFHDQPHLKRKAFCSFQIYAVPWQGTMTLSKSTCQDA     146
|       |       ||||  |||   | || |  |     | ||
KPLSTNEICAIQENSKLKRKLSCSFLVGALPWNGEFTVMEKKCEDA     142
```

A PHI-BLAST search given the CYTC_HUMAN sequence and the common motif identified by eMotif (Huang *et al*, 2001) does indeed find the CST8_HUMAN sequence and generates a local alignment. (In PHI-BLAST the matched pattern is indicated by asterisks.)

```
Score = 54.1 bits (150), Expect = 7e-16
 Identities = 39/108 (36%), Positives = 61/108 (56%), Gaps = 2/108 (1%)

Query:  39  PMDASVEEEGVRRALDFAVGEYNKASNDMYHSRALQVVRARKQIVAGVNYFLDVELGRTT 98
pattern 53                *************
            P++AS     V++ L FA+ EYNK S D Y    ++ ++A+ Q+   + Y +DVE+ R+
Sbjct:  37  PVNAS--NANVKQCLWFAMQEYNKESEDKYVFLVVKTLQAQLQVTNLLEYLIDVEIARSD 94

Query:  99  CTKTQPNLDNCPFHDQPHLKRKAFCSFQIYAVPWQGTMTLSKSTCQDA 146
            C K     + C   +   LKRK  CSF + A+PW G  T+ +  C+DA
Sbjct:  95  CRKPLSTNEICAIQENSKLKRKLSCSFLVGALPWNGEFTVMEKKCEDA 142
```

PHI-BLAST matches the common pattern and builds the rest of the alignment around the match. Note that PHI-BLAST inserts the same gap in CST8_HUMAN as the unforced affine global alignment. This is because the regular expression motif used for the PHI-BLAST search is shorter than the block definition, and the location where the gap is placed lies within the block, but outside the regular expression.

This paper will take as a working assumption the supposition that the better a block's E-Value, the less appropriate it is to place a gap within it. The E-Value of the eMatrix (Wu, *et al*, 1999) block used here, 8.75 e-02, is good enough to suggest that alignments which preserve the block should be favored. However, much better E-Values are sometimes seen, and this one may not be high enough to require a strict block match.

Another Example

Comet and Henry use an example from the PROSITE family PS01288, which has four members in Swissprot release 35. They focus on a pairwise alignment between two members, RTCB_ECOLI and Y682_METJA, which share several patterns that normal Smith-Waterman does not correctly align.

A search of the eBlocks database did not reveal any shared blocks or regular expressions between the two sequences, probably because the family is too small to merit inclusion in the database. However, a search of the Blocks+ database (Henikoff, *et al*, 1999) did find a

family, IPB001233, which corresponds to PROSITE family PS01288 and contains both sequences. The family has 5 blocks, all of which are found in both sequences:

	RTCB_ECOLI		Y682_METJA	
Block Name	Start – End	E-Value	Start - End	E-Value
IPB001233A	29 – 64	2.4e-20	46 – 81	1.8e-30
IPB001233B	65 – 78	1.1e-06	85 – 98	1.8e-11
IPB001233C	159 – 172	7.4e-09	682 – 695	1.5e-09
IPB001233D	176 – 215	9.5e-22	713 – 752	1.4e-28
IPB001233E	216 – 247	5e-15	753 – 784	4.9e-26

In some respects, this example seems like a poor choice for the current program. Y682_METJA is a much longer sequence than RTCB_ECOLI, so a global alignment will never have a high score. Nonetheless, the sequences and blocks were entered into the program and an alignment was run.

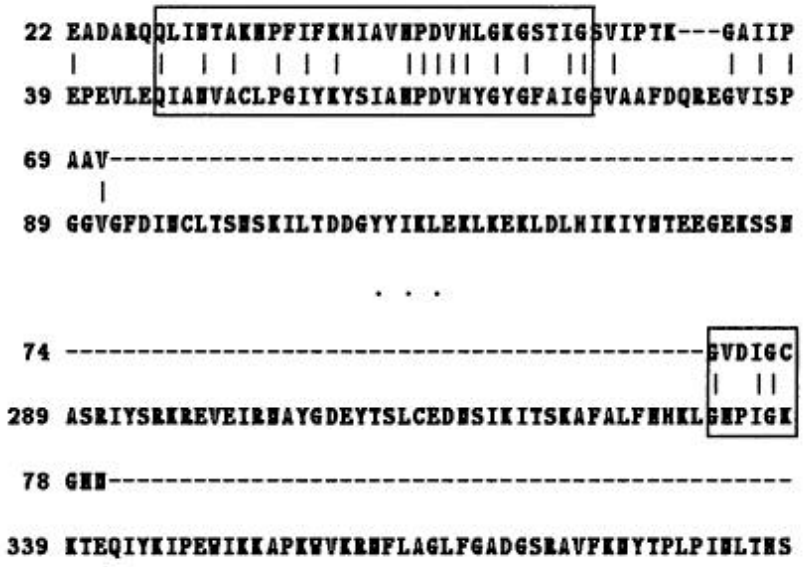
Surprisingly, the block alignments were very close. The last three blocks, located near the end of both proteins, aligned perfectly. The second block, near the beginning, also aligned perfectly. This agreement between the four blocks lends credence to the otherwise dubious global alignment. However, there was a small problem with the first block. A gap was inserted near the end of the first block in the RTCB_ECOLI sequence that caused a mismatch with the corresponding block in Y682_METJA. Since the proteins are so long, the full alignment is left to an appendix, however the troublesome spot is shown below.

```
                                                  11111
MNYELLTTEN--------------APVKMWTKGV---PVEADARQQLINT 33
|   |                   |             |     |  |
MKDVLKRVSDVVWELPKDYKDCMRVPGRIYLNEILLDELEPEVLEQIANV 50
                                                  11111

11111111111111111111111111111   11122222222222222
AKMPFIFKHIAVMPDVHLGKGSTIGSVI---PTKGAIIPAAVGVDIGC-- 78
|  | | |    ||||| | |  || |       | | |  || || |
ACLPGIYKYSIAMPDVHYGYGFAIGGVAAFDQREGVISPGGVGFDINCLT 100
11111111111111111111111111111111   22222222222222
```

Since all the other blocks match perfectly, and the E-Value for this block is an exceptional 2.4 e-20, it seems reasonable in this case to force the blocks to align. Calculations from the Blosum50 matrix indicate that this will only lower the total score by 3 points. (Scores shown with the force blocks option are for the particular subsequence.)

```
BEFORE BLOCK:
Score = -70
MNYELLTTEN--------------APVKMWTKGV---PVEADARQ      28
|   |                    |             |
MKDVLKRVSDVVWELPKDYKDCMRVPGRIYLNEILLDELEPEVLE      45


BLOCK:
Score = 102
QLINTAKMPFIFKHIAVMPDVHLGKGSTIGSVIPTK               36
|  | |  | | |    ||||| | |  || |
QIANVACLPGIYKYSIAMPDVHYGYGFAIGGVAAFD               36

BEFORE BLOCK:
Score = -24
---                                                0

QRE                                                3

BLOCK:
Score = 58
GAIIPAAVGVDIGC                                     14
| | |  || || |
GVISPGGVGFDINC                                     14
```

Direct comparison to SWP is difficult, because it fits a number of small patterns that are not reflected in the block databases. However, SWP does use a pattern corresponding closely to Block 1 above and a smaller pattern comes into play in the area covered by Block 2. This portion of the SWP alignment appears below. (It was carefully cut from an electronic copy of the paper using Adobe Acrobat.)



The two patterns matched by SWP have rectangles drawn around them. The first pattern corresponds to Block 1, but is 6 residues shorter. Nonetheless, the 6 residues following the pattern do align in SWP exactly the way they do in the forced block alignment.

Block 2 is more problematic. The first part of Block 2 is not represented by a pattern in SWP, but it begins immediately after the first gap in RTCB_ECOLI (the top sequence) and continues on in correct alignment for 8 residues. Then something unusual happens. Where both the unforced global affine alignment and the forced alignment continue matching Block 2, SWP inserts a very long gap in RTCB_ECOLI, so that it can match the next 6 residues to a small pattern over 200 residues away in Y682_METJA. The E-value of Block 2 in RTCB_ECOLI is a respectable 1.1e-06, so splitting the block down the middle to match such a distant pattern seems hard to justify.

The ability to introduce long gaps to connect distant similarity regions is a feature of SWP controlled by the weight given to the patterns. Unfortunately, the above example may indicate that this feature is a two-edged sword. The weights that give SWP the ability to reach across long gaps to match known patterns may also give it the ability to overlook unknown patterns, even when they are very close. In this case, continuing to match Block 2 would have resulted in more identities than matching the far off pattern. Since Block 2 was not presented to SWP as a regular expression pattern, matching it became less rewarding than inserting a 200 residue gap to match a known pattern.

The difficulty of choosing the appropriate weights for patterns is well known to the authors of SWP, and is cited as an area in need of further work. The above example is therefore not an indictment of the whole approach, but merely a reason for careful application of the algorithm. In this case, the availability of an alignment program incorporating block information serves as a useful check on the SWP alignment.

Discussion

The prototype pairwise sequence alignment program presented here is similar to existing pattern-aware alignment programs with some important differences. It differs primarily in its use of blocks rather than regular expressions, and in its use of a global alignment algorithm rather than local alignment. It also offers the option of merely showing block alignments rather than forcing them, which can be useful in evaluating traditional alignments.

The use of blocks instead of regular expressions has some advantages. It avoids the problem of making alignments within variable length matching patterns that affects SWP. Also, it takes good advantage of the publicly available blocks databases such as

BLOCKS+, which appear to be more comprehensive than many regular expression databases.

The prototype program effectively brings the information now available in block databases to bear on pairwise sequence alignments. It makes it easy for the researcher to confirm the alignment of matching blocks or notice discrepancies that might deserve further investigation. And, in some cases, it can serve as a useful check on other alignment methodologies.

The addition of block information to the pairwise alignment process is not a silver bullet that instantly resolves all alignment problems. However, it does bring a new dimension to the task that can be helpful in assessing and guiding alignments. For this reason, it could be a useful addition to the researcher's toolkit, especially when used to complement existing alignment programs that use regular expressions.

References

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25,** 3389-3402.

Comet, J.-P., Henry, J. (2002) Pairwise sequence alignment using a PROSITE pattern-derived similarity score. *Computers and Chemistry* **26,** 421-436.

Cornwall, G.A., Hsia, N., Sutton, H.G. (1999) Structure, alternative splicing and chromosomal localization of the cystatin-related epididymal spermatogenic gene. *Biochem. J.* **340,** 85-93.

Durbin, R., Eddy, S., Krogh, A., Mitchison, G. (1998) *Biological Sequence Analysis.* Cambridge University Press.

Henikoff, S., Henikoff, J.G., and Pietrolovski, S. (1999) Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. *Biochemistry* **15**, 471-479.

Huang, J.Y. and Brutlag, D.L. (2001) The eMotif database. *Nucleic Acids Res.* **29**, 202-204.

Rawlings, N.D., Barret, A.J. (1990) Evolution of proteins in the cystatin superfamily. *J. Mol. Evol.* **30**, 60-71.

Sestoft, P. Programs for Biosequence Analysis (1999) http://www.dina.kvl.dk/~sestoft/bsa.html

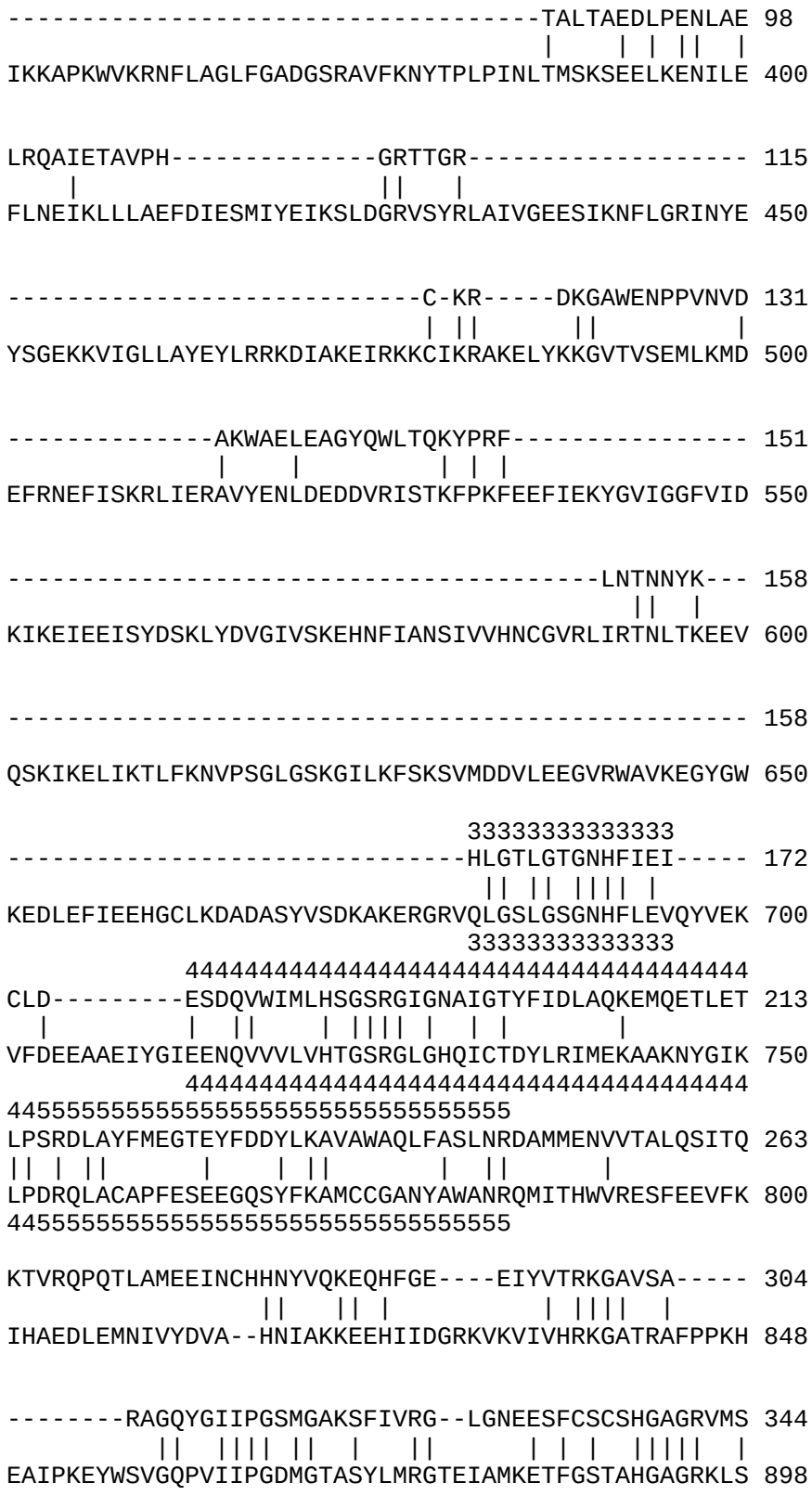
Su, Q., Lu, L. and Brutlag, D. (2002) http://fold.stanford.edu/eblocks/

Wu, T.D., Nevill-Manning, C.G., and Brutlag, D.L. (1999) Minimal-risk scoring matrices for sequence analysis. Journal of Computational Biology **6**, 219-235.

Appendix – The full alignment of RTCB_ECOLI and Y682_METJA

	RTCB_ECOLI		Y682_METJA	
Block Name	Start – End	E-Value	Start - End	E-Value
IPB001233A	29 – 64	2.4e-20	46 – 81	1.8e-30
IPB001233B	65 – 78	1.1e-6	85 – 98	1.8e-11
IPB001233C	159 – 172	7.4e-09	682 – 695	1.5e-09
IPB001233D	176 – 215	9.5e-22	713 – 752	1.4e-28
IPB001233E	216 – 247	5e-15	753 – 784	4.9e-26

```
Gap Penalty 16 Gap Extension Penalty 4 Blosum 50 Matrix

AFFINE GLOBAL:
Score = -1638
                                                  11111
MNYELLTTEN--------------APVKMWTKGV---PVEADARQQLINT 33
|   |                   |             |     |  |
MKDVLKRVSDVVWELPKDYKDCMRVPGRIYLNEILLDELEPEVLEQIANV 50
                                                  11111
11111111111111111111111111111   11122222222222222
AKMPFIFKHIAVMPDVHLGKGSTIGSVI---PTKGAIIPAAVGVDIGC-- 78
|  | | |    ||||| | |  || |       | | |  || || |
ACLPGIYKYSIAMPDVHYGYGFAIGGVAAFDQREGVISPGGVGFDINCLT 100
11111111111111111111111111111   22222222222222

---------------------------------------------------- 78

SNSKILTDDGYYIKLEKLKEKLDLHIKIYNTEEGEKSSNILFVSERYADE 150


---------------------------------------------------- 78

KIIRIKTESGRVLEGSKDHPVLTLNGYVPMGMLKEGDDVIVYPYEGVEYE 200


---------------------------------------------------- 78

EPSDEIILDEDDFAEYDKQIIKYLKDRGLLPLRMDNKNIGIIARLLGFAF 250


-----------------------------------GMNALR-------- 84
                                     |  | |
GDGSIVKENGDRERLYVAFYGKRETLIKIREDLEKLGIKASRIYSRKREV 300


---------------------------------------------------- 84

EIRNAYGDEYTSLCEDNSIKITSKAFALFMHKLGMPIGKKTEQIYKIPEW 350
```



```
RTKAKKLFSVEDQIRATAH--VECRKDAEVI--DEIPMAYKDIDAVMAA- 389
| || ||       |  |        |       | | |||  | |
RAKALKLWKGKEIQRRLAEMGIVAMSDSKAVMAEEAPEAYKSVDLVADTC 948


-QSDLVEVIYTLRQVVCVKG                               408
            |     ||
HKAGISLKVARMRPLGVIKG                               968
```