Notes on using find_max_cover

Purpose:

find_max_cover takes as input a list of local alignments, identified by their starting positions and lengths in a query sequence. From this information, the program identifies the combination of alignments that either **a)** covers the maximum number of residues in the query sequence or **b)** has the maximal alignment score. The user can allow two alignments to overlap up to a desired threshold. The program also has the ability to compute the expectation value (E-value) for combinations, given information about the size of the query sequence and the size of the database of sequences it was queried against. A manuscript on the algorithm used is in progress.

Installation:

To build under Linux or other platforms with g++, type

```
% tar xvf find_max_cover.tar
% cd find_max_cover
% make
```

You make need to modify the makefile for platforms without g++.

Usage: `find_max_cover <filename> (-o:<number>) (-e) (-s)`

Arguments:

The program accepts four arguments. The first one must be the name of a file that contains the alignment positions (the format of this file is described below).
The remaining three optional parameters are case-insensitive and preceded by a dash.

The –o:# option allows a maximum of # residues to overlap in the combinations of alignments. If the –o option is not specified, no overlap is permitted (equivalent to –o:0). # must be an integer greater than or equal to zero.

The –e option causes the program to compute E-values (See Altschul and Gish, 1996 *Methods in Enzymology*, **266**:460-480) for the combinations found. Using –e requires additional information to be included in the alignment position file. See the description of that file below.

The –s option causes the program to search for the alignment combination with the largest alignment score rather the longest combination. (The alignment score is the sum of match, mis-match, and gap penalties for the alignment in question). This option also requires specifying further information in the input file.

Alignment file:

The most basic format for this file is:

```
Begin Alignments
<Align Name 1> <Start> <Length>
<Align Name 2> <Start> <Length>
                .
                .
                .
```

The “Alignment Name” field can contain any desired text so long as it does not include spaces and is unique in the file. Starting position and length should both be integers. Note that lines of whitespace and those beginning with “#” are ignored.

If you are calculating E-values or using scores are you optimality criterion, you will need to add a score field thus:

```
Begin Alignments
<Align Name 1> <Start> <Length> <score>
<Align Name 2> <Start> <Length> <score>
              .
              .
              .
```

If, in either of these cases you have set the overlap parameter to be non-zero, you will need to include alignment locations:

```
Begin Alignments
<Align Name 1> <Start> <Length> <score> <alignment file>
<Align Name 2> <Start> <Length> <score> <alignment file>
              .
              .
              .
```

These alignment files (“alignment file” field) can be in PIR, FASTA, NEXUS, or PHYLIP format, and must end in .pir, .fas, .nex or .phy to indicate the respective format. The alignment files are required because the presence of overlap between alignments may require that the alignment scores to be adjusted.

Example files for various option settings are provided on our website (http://www.unm.edu/~compbio/software/find_max_cover).

- Standard case, no E-values (example_noeval.txt)
- E-values requested, overlap set to 0 (example_eval_o0.txt)
- E-values with non-zero overlap (example_eval_overlap.txt)
- Score optimality criterion with non-zero overlap (example_score.txt)

Options in the alignment file:

When calculating E-values or using alignment scores as the optimality criterion, there are several addition pieces of information that may be necessary. For convenience, many of these parameters can take on default values if not specified. The table below lists the options, their defaults and any other relevant information. Options are case-insensitive.

Option	Default	Used With	Details
<protein>/<nucleotide>	Nucleotide	Scores/E-values	Sequence type
K = #[a]	0.138	E-values only	Default values most likely invalid
L = #[a]	0.6	E-values only	Default values most likely invalid
H = #	0.449	NA	Currently ignored
Match = #	5	Scores/E-values; *overlap* ≠ 0	Sets the match score for nucleotide alignments
Mismatch = #	-4	Scores/E-values; *overlap* ≠ 0	Sets the mis-match score for nucleotide alignments
Matrix file = <filename>	Hard-coded BLOSUM62 matrix	Scores/E-values; *overlap* ≠ 0	format of the matrix file should match http://www.unm.edu/~compbio/software/find_max_cover/blosum62.bla
Gap open = #[b]	-12	Scores/E-values; *overlap* ≠ 0	Same for protein and nucleotide alignments
Gap extend = #[b]	-2 (protein), -4 (nucleotide)	Scores/E-values; *overlap* ≠ 0	
Mlen = #[c]	None	E-values only	Length of query sequence
Nlen = #[c]	None	E-values only	Length of reference sequence

[a]: *K* and *Lambda* are functions of the alignment scoring matrix used. Programs such as BLAST have tables giving *K* and *Lambda* values for different alignment matrices and gap penalties. *H* is an entropy parameter that is not currently used. These are needed to calculated E-values even when overlap is 0.

[b]:Gap open and extension penalties are specified in what appears to be a standard way. Thus the penalty for a gap of one character is <Gap open> while that for a gap of three characters is <Gap open> +2*<Gap extension>. Only needed when *overlap* ≠ 0

[c]:The effective query and database length (second line) can be calculated if the values of *K*, *Lambda* and *H* are known (see Altschul and Gish, 1996 above). Using the actual length of the query or database instead of effective lengths may decrease the accuracy of the computed E-value. These are needed to calculated E-values even when overlap is 0.

Example Input and output:

The output (directed to standard output) consists of a listing of the number of residues aligned by the alignment combination, the list of alignments contained in that combination, the number of alignments in that combination, and the E-value for the combination (if requested).

- Criterion: Longest combination. No E-values, no overlap:

```
% find_max_cover example_noeval.txt

Settings are:
Nucleotide sequence.
Best combination aligns 131 residues
Best alignment combination: A0  A4
Combination contains 2 alignments
```

- Criterion: Longest combination. E-values but no overlap:

```
% find_max_cover example_eval_o0.txt –e

Calculating E-values for combinations
Settings are:
Query size: 333 Reference sequence/DB size: 7177762
kappa: 0.138 Lambda: 0.6 H: 0.449
Protein sequence.
Best combination aligns 166 residues
Best alignment combination: A0  A4      A5
Combination contains 3 alignments
E-value for combination is 8.36225e-147
```

- Criterion: Longest combination. E-values with overlap

```
% find_max_cover example_eval_overlap –e –o:10

Calculating E-values for combinations
Settings are:
Query size: 333 Reference sequence/DB size: 7177762
kappa: 0.138 Lambda: 0.6 H: 0.449
Protein sequence. Using default BLOSUM62 matrix
Gap opening penalty: -12 Gap extension penalty: -2
Best combination aligns 16 residues
Best alignment combination: Align0      Align1
Combination contains 2 alignments
E-value for combination is 0.149763
```

- Criterion: Largest Score. With overlap

```
% find_max_cover example_score_calc_overlap.txt –s –o:10

Settings are:
Protein sequence. Using default BLOSUM62 matrix
Gap opening penalty: -12 Gap extension penalty: -2
Best combination has a score of 69
Best alignment combination: Align0      Align1
Combination contains 2 alignments
```

Source code information

find_max_cover is written in c++ and has been compiled with Visual C++ 6.0 for Windows and gnu g++ version 2.96 for Linux. Because the program uses templates, it may not compile under all c++ compilers (Templates are a relatively new addition to the language). In general, compiling templates seems to work best if the file describing the template is #included into the file using it, rather than separately compiled. An executable for Win9x/WinNT/Win2000/WinXP is available from our website.