ERPIN

Easy RNA Profile IndentificatioN

DOCUMENTATION

Version 5.5 (C)2001-2006

Daniel Gautheret: gautheret@esil.univ-mrs.fr

André Lambert: lambert@cpt.univ-mrs.fr

Please cite:

ERPIN algorithm : Gautheret D. & Lambert A. (2001). Direct RNA definition and identification from multiple sequence alignments using secondary structure profiles. *J. Mol. Biol.* 313:1003-1

ERPIN E-values: Lambert A, Legendre M, Fontaine JF, Gautheret D. (2005) Computing Expectation Values for RNA Motifs Using Discrete Convolutions. *BMC Bioinformatics*. **6**(1):118

ERPIN Web server: Lambert A., Fontaine JF, Legendre M, Leclerc F, Permal E, Major F, Putzer H, Delfour O, Michot B & Gautheret D (2004) The ERPIN server: an interface to profile-based RNA motif identification. *Nucl Acids Res*. **32**, W160-5.

Erpin web site: http://tagc.univ-mrs.fr/erpin/ (for downloads and online searches)

1. THE ERPIN COMMAND LINE

```
========== erpin: Easy Rna Profile IdentificatioN, Version 5.3 ===========

Usage:
erpin [-h]                                                          help
erpin <training-set>                                  training set file name
      <input-file>                                database file name (fasta)
      <region>                                          region of interest
      -nomask|((-mask|-umask|-add) <elt1> ...)      level1,
      [-nomask|((-mask|-umask|-add) <elt1> ...)]    level2,     default: void
      [...]                                         level..              idem
      [-cutoff <cutoff1> <cutoff2> ..]                          default: 100%
      [-dmp|-smp]                                               default: -dmp
      [-fwd|-rev|-fwd+rev]                                  default: -fwd+rev
      [-long|-short|-mute]                                     default: -long
      [-warnings]                                                default: OFF
      [-globstat|-locstat|-unifstat]                       default: -globstat
      [-Eon|-Eoff]                                E-value ON|OFF, default: -Eon
      [-hist]                                                    default: OFF
      [-seq1 <seqnb1>][-nseq <nseq>]             default: 1, SEQ_MAX_NB (all)
      [-bgn <seqbgn>][-len <range>]                   default: 1, SEQ_MAX_LEN
      [-logzero <logzero>]                                       default: -20
      [-tablen <tablen>]                                        default: 1024
      [-chrono]                                                  default: OFF
      [-pcw <pcw>]                           pseudo-counts weight, default: 0.1
      [-hpcw <hpcw>]             pseudo-counts weight for helices, default: 0.1
      [-spcw <spcw>]             pseudo-counts weight for strands, default: 0.1
      [-sumf <fname>]   file of alternate user defined substitution matrices

default command line: "erpin <trset> <data> <region> <mask>" stands for:

erpin <trset> <data> <region> <mask> -cutoff 100% \
      -dmp -fwd+rev -long -globstat -Eon \
      -seq1 1 -nseq 10M -bgn 1 -len 300M -pcw 0.1 -logzero -20. -tablen 1024

Example:
erpin trsets/trna.epn  sequences/ecoli.fna -2,2 -umask 20 11 -nomask \
      -cutoff 100% 90% -sumf ../sum/SUM.dat -fwd -short -chrono

============= D.Gautheret A.Lambert, Luminy Marseille, Mar 2006 ============
```

2. COMPULSORY ARGUMENTS

Database file

The sequence database file is in the FASTA format. The file can hold up to 10 million sequences, each less than 300Mb long.

Training set: the .epn file

Training sets are FASTA-like files starting with a secondary structure descriptor that encodes each structure element with a user-defined number. Numbers that are repeated at two different places stand for helical elements, other numbers stand for single strands. Two or three numbering lines can be used if necessary. Element numbers do not have to be continuous or ordered, but zero is not a permitted number.

Sequences should be aligned with gaps, so that all entries have the same length as the secondary structure descriptor.

- No gaps are tolerated within helical regions.

- Both T and U are accepted.

- Both uppercase and lowercase are accepted.

Training set example (tRNA):

```
>structure
00000000000000000000000000000000000000000000000000002222211111112222200000001111
12222222334444555555555554444677777888888877777999990000011111110000022222222222
>DA0260
-GGGCGAAUAGUGUCAGC-GGG--AGCACACCAGACUUGCAAUCUGGUAG-GGAGGGUUCGAGUCCCUCUUUGUCCACCA
>DA0340
-GGGCUCGUAGCUCAGC--GGG--AGAGCGCCGCCUUUGCGAGGCGGAGGCCGCGGGUUCAAAUCCCGCCGAGUCCA---
>DA0380
-GGGCCCAUAGCUCAGU--GGU--AGAGUGCCUCCUUUGCAAGGAGGAUGCCCUGGGUUCGAAUCCCAGUGGGUCCA---
>DA0420
-GGGCCCAUAGCUCAGU--GGU--AGAGUGCCUCCUUUGCAAGGAGGAUGCCCUGGGUUGGAAUCCCAGUGGGUCCA---
```

In this training set file, elements 02, 04, 07 and 20 describe the four tRNA helices, other numbers describe single strands.

A Perl script (**`parent2epn.pl`**) is provided to convert a parenthesized (Fasta-like) alignment into .epn format. This script will also check (and optionnally correct) gaps in helices and gap-only columns.

Limitations: Training sets should not exceed 12,000 sequences. Helices should not exceed 64bp.

Regions and Masks (-mask, -add)

Compulsory argument `<region>` contains two comma-separated numbers `<r1>,<r2>` defining the boundaries of the region of the alignment that will be used for searches. These numbers refer to the structure header of the training set file. When a boundary is a helix, use "plus" or "minus" signs to specify 5' or 3' strand, respectively. The "plus" sign is optional. Output sequence alignments show the defined region only. When a region contains only one strand of a helix, this strand is treated like a single-stranded element during search.

Another compulsory argument is the Mask. When `-nomask` is used, the whole region is searched. Other types of masks are exposed later.

Examples (using the tRNA training set above):

```
erpin trna.epn coli.fasta -4,4 -nomask
```

-> Search region includes helix 4 and strand 5

```
erpin trna.epn coli.fasta -4,11 -nomask
```

-> Search region includes helix 4 and 7, strands 5, 8 and 9, and the 5' part of stem 20

```
erpin trna.epn coli.fasta -2,2 -nomask
```

-> Search region includes the whole alignment except for strands 1 and 12

Masks are used to restrict searches to certain elements in a region. A mask is followed by numbers indicating which elements are included or excluded. Masks do not use plus or minus signs for helices. When a helix is refered to, both strands are used.

Types of masks:

`-mask i j .. n`	: elements i,j..n are excluded
`-umask i j .. n`	: elements i,j..n are included
`-add i j .. n`	: elements i,j..n are added to mask (multi-level search)
`-nomask`	: all elements of the region are included

CAUTION: with `-nomask`: When a region is large or contains several gapped strands, this may result in huge memory and CPU usages. Use multi-level searches in this case (see below).

Examples using the tRNA training set above:

```
erpin trna.epn coli.fasta -2,+2 -mask 8
```

-> Searches region -2 to +2 ignoring strand 8

```
erpin trna.epn coli.fasta -2,+2 -umask 2 4 7
```

-> Searches only elements 2, 4 and 7 in region -2 to +2

Multi-level search strategies

This is done by applying several masks consecutively. When the command line contains several masks, Erpin will conduct a first search using the first mask, and continue with the next mask only if a solution above cutoff has been found with the previous mask. Since the search at level n is performed only around solutions found at level n-1 (within distance intervals specified in the input alignment), the search speed is increased. For instance, the following command:

```
erpin trna.epn coli.fasta -2,+2 -umask 20 11 -nomask
```

will run much faster than:

```
erpin trna.epn coli.fasta -2,+2 -nomask
```

.. and yet produce similar result. This permits to implement multi-level search strategies, where the most significant signatures are searched first and the motif is gradually expanded thereafter, thus speeding up database searches. Any number of consecutive masks are allowed. By default, the score cutoff for each mask is 100%. This can be changed using the cutoff argument, as follows:

```
erpin trna.epn coli.fasta -2,+2 -umask 20 11 -nomask -cutoff 10 20
```

-> in this example, a cutoff of 10 is used at the first level and a cutoff of 20 at the second level.

Use " `-nomask` " as the last step if you want to display the complete alignment. Bear in mind however that " `-nomask` " on a long, gapped region may cause memory/CPU overflow!

Another important thing to understand about multi-level searches: In the following command:

```
erpin <tr-set> <databank> 1,10 -umask 1 2 -umask 3 4 -umask 5 6
```

the last step of the search only seeks elements 5 and 6. Although elements 1, 2, 3 and 4 have been detected at previous steps, they are not considered anymore at this stage and will not be included in the final output (in this case, the previous mask level serves only for setting search space boundaries for the next mask level). Generally, users will prefer constructing masks incrementally, so that the whole region is matched in the end, such as:

```
erpin <tr-set> <db> 1,10 -umask 1 2 -umask 1 2 3 4 -umask 1 2 3 4 5 6
```

To simplify such a command line, one may use the `-add` parameter. " `-add X` " just updates the previous mask by unmasking element X from it. Therefore, the previous command can be written:

```
erpin <tr-set> <db> 1,10 -umask 1 2 -add 3 4 -add 5 6
```

If the higher search level still contains masked elements (like elements 7 to 10 above), the sequences within these elements will appear UNALIGNED in the final output (left-justified in a space of same size as in training set). Unaligned sequences are shown in lowercase.

A good general strategy could use 3 levels:

- first level to speed up search by selecting a short motif

- second level for specificity: extend to a larger motif eliminating false positives

- third level with the complete region unmasked for output.

However, you won't handle a 16S RNA with just 3 levels!

Beware: a 2-level search using a first mask at cutoff X and then the entire region (nomask) at cutoff X is not equivalent to a single level search for the entire region at cutoff X. Indeed, a solution of score X may have a lower score in any of its part. Then, this part would be missed at the first level using a cutoff of X. ALWAYS use a lower cutoff for the first stage.

3. OPTIONAL ARGUMENTS

Pseudocounts (-pcw)

Training sets with few sequences (i.e.20 or less) result in hollow weight matrices that strongly penalize any variation from training set sequences and thus affects the sensitivity of detection (see also "log zero"). To solve this problem, ERPIN uses pseudocounts (from version 4.2). Pseudocounts introduce some articifical base or base-pair counts in the weigth matrix, that simulate what could have been observed in a larger sequence alignment. Pseudocounts require some prior knowledge of "typical" mutation frequencies in RNA molecules. Let's pretend that G often mutates to A in typical RNAs. Then if a column of the initial alignment mostly contains Gs, we can expect that some As should occur too. This is the way pseudocounts work. "Typical" mutation rates for single-strands and base-pairs were evaluated from a 16S/18S ribosomal RNA alignement, and the detailed counting procedure resembles that of Henikoff and Henikoff (CABIOS 1996, 12:135).

Users can set the level of pseudocount to be injected in ERPIN profiles, or pseudocount weight. A high pseudocount weight is necessary when training sets are really poor, but this will affect search specificity. In the extreme situation where profiles are made of 100% pseudocounts, ERPIN searches would just produce noise.

Pseudocounts weights are set using the **`-pcw`** parameter. Internally, pseudocount weights are comprised between 0 and 1 with a default value of 2x10e-3. At weight=0 no pseudocount is used and at weight=1, profiles are 100% pseudocounts. For reasons of backward compatibility, user values for pseudocount are different. **Caution: since V.5.0: user values for pseudocounts range between 0 and 500** The default value of **`-pcw`** is 0.1 (corresponding to an internal pseudocount weight of 2x10e-3).

Recommended pseudocount weights. Non expert users should leave -pcw at default value. Expert users may use larger weights (for instance 10, 100, 400) for small training sets, always keeping in mind that a higher pcw may produce more false positives. For instance, a training set containing only 4 correctly aligned tRNA sequences can retrieve 80% of E. coli tRNAs using **`-pcw 400`**, with two false positives. **Important:** To fully benefit from a higher pcw, users should relax score cutoffs at the same time, using the **`-cutoff`** parameter (we are working on automatizing this task).

Score cutoff (-cutoff)

Optional argument **`-cutoff`** is used to set the lower limit of hit scores. Erpin uses two types of scores: absolute and relative.

Absolute scores are the sum of scores obtained for each secondary structure element in the current mask, obtained directly from the lod-score profiles. An integer or real number following the **`-cutoff`** tag is interpreted as an absolute score, unless followed by a "%" sign. To get help in establishing absolute score cutoffs, visualize absolute scores for the training set sequences using the **`tstat`** program (see Extra Tools Section).

Relative scores are expressed as a percentage of training set sequences captured. Theses scores are followed with a "%" sign. A score of 50% for a given region or mask is a score capturing 50% of sequences in the training set for this region or mask. A score of 100% is the lowest score in the training set for this region or mask. The default score for any region or mask is 100%.

Examples:

```
erpin trna.epn coli.fasta -2,+2 -nomask -cutoff 20
```

-> Select hits with a absolute score higher than 20

```
erpin trna.epn coli.fasta -2,+2 -nomask -cutoff 90%
```

-> Select hits with a score higher than that of 90% of training set sequences

When several mask levels are used, a cutoff is needed for each level. Scores must then be ordered exactely as masks., e.g.:

```
erpin trna.epn coli.fasta -2,2 -umask 4 -add 5 -nomask -cutoff 0 10 15
```

-> Here, the score cutoff for first mask level (**`-umask 4`**) is 0, score cutoff for second mask level (**`-add 5`**) is 10 and score cutoff for third mask level (**`-nomask`**) is 15. In case of missing cutoffs, cutoffs for all non-assigned mask levels are set to 100%.

Since version 5.3, score cutoffs higher than 100% are permitted. This corresponds to cutoffs that are low enough to capture RNAs scoring less than the worst sequence in the training set. How do we know which cutoff could capture, say, 105% of a training set? This is done by extrapolating the score distribution for this region or mask, using a polynomial function. Since scores usually drop sharply after 100%, extrapolated cutoffs are limited to 110%. Any cutoff higher than 110% will be automatically reset to 110%.

(see also sections "log-zero" and "expected frequencies")

E-value (-Eon, -Eoff)

Since Version 3.9, ERPIN computes Expect-values (E-values). For any hit of score S obtained in a given database, The E-value is the number of hits of same or higher score that can be expected by chance in the same database. It thus provides user with a statistical significance of hits. Typically an E-value of 10e-2 or less is significant. However, many non-biological hits might arise with "good" E-values, due to other factors such as the presence of a low complexity region in the motif.

E-value calculation is turned on by default. It can be turned off using the **`-Eoff`** parameter (this may be useful for training sets containing large number of gaps that take a toll on CPU time.

Reverse complement searches (-fwd, -rev)

Searches are performed on both strands by default. Use -fwd or -rev to limit search on plus or minus strand, respectively.

Output format (-long, -short, -mute)

Output format can be set as long, short or mute.

`-long`: for every hit, prints scores at each search level, coordinates and complete sequence.

`-short`: for every hit, prints coordinates and final score.

`-mute`: only prints final number of hits.

Selection of database sequences (-seq1, -nseq, -bgn, -len)

Database searches can be restricted to subsets of sequence files, specified as follows:

`-seq1 n`: search begins at sequence number n in the database

`-nseq n`: n sequences in the database are searched

`-bgn n` : search starts at position n in each sequence

`-len n` : n nucleotides are searched after start point

Elapsed CPU time (-chrono)

The **`-chrono`** argument displays CPU time used for the search phase.

4. ADVANCED OPTIONS

Dynamic/Static Mask Processing (-dmp / -smp)

By default, multiple-level searches imply that each level starts with the partial configurations found at the previous level. In the last example above, the search for elements 2 & 3 at step 2 will be performed for each configuration of elements 1 and 2 identified at step 1. This is called "Dynamic Mask Processing", and corresponds to the **`-dmp`** option.

With the -smp option (Static Mask Processing), each search step is performed independently. In most cases, this results in higher CPU times, since all elements in the current mask must be identified at the same time. A possible use of `-smp` is to recover some "lost" solutions by lowering score cutoffs in late stages.

Expected/background nucleotide frequencies (-locstat, -globstat, -unifstat)

Single strand and helix profiles are computed based on observed and expected frequencies for each base and base pair. By default, expected frequencies are those in the search database, averaged over all sequences (`-globstat` option). If the database is very heterogeneous (e.g. a mixture of sequences from different organisms), expected frequencies should be computed independently for each sequence, using the " `-locstat` " option. Beware however, that short sequences may have highly biased compositions, possibly resulting in spurious high-scoring solutions. The " `-unifstat` " (uniform) option sets A/T/G/C frequencies to 0.25 each.

Advanced pseudocounts (-hpcw, -spcw, -sumf)

Expert users may want to separately set pseudocount weights for helices and single strands. Parameters `-hpcw` (helices) and `-spcw` (single strands) will do just this.

The `SUM.dat` file in the ERPIN distribution contains the default substitution matrices used for pseudocount calculation. There is one 16x16 matrix for base-pairs and one 4x4 matrix for single strands. Users can provide an alternate subsitution matrix file using the `-sumf` parameter. Base order is A,T,G,C in the 4x4 matrix and A:A,A:T,A:G,A:C,T:A,T:T,etc. in the 16x16 matrix. Example:

```
erpin <tr-set> <databank> 1,10 -nomask -sumf MYSUM.dat
```

The `mksum` program provided in the distribution constructs new subsitution matrices from any alignment.

Log zero (-logzero)

In the absence of pseudocounts, the lod-scores of bases or basepairs that are never observed at a given position are set at the default arbitrary value of -20. The " `-logzero` " parameter can change this. A higher logzero (e.g. -5) will result in a greater tolerance to deviations from the training set sequences.

The logzero parameter is irrelevant when pseudocounts are used (default behavior)

Length of pre-computed alignements (-tablen)

Erpin precalculates alignments of single stranded elements over a sliding window of 1024 nt. This can be changed using the `-tablen` argument. Raising this number could save a little CPU time when motifs longer than 100 nt are searched.

Histogram (-hist)

Creates histogram of scores obtained in the current search. Generates a file named `epnhist.dat`, the first line of which contains:

- lowest score in current search (minus epsilon)
- highest score in current search (plus epsilon)

- number of score intervals in histogram
- total number of solutions

This is followed by histogram values (one integer per interval).

The **`-hist`** option is best used in conjunction with the **`epnstat`** program to evaluate score distributions in random sequences.

5. EXTRA TOOLS

Extra tools are provided in the **`/apps`** and **`/tools`** directories. We describe here the most useful programs.

parent2epn.pl

This Perl script reads a parenthesized alignment (fasta-like format) and translates it into .epn (ERPIN2) format. The program also checks that helices have no gap (otherwise change positions to single-stranded) and deletes columns with gaps only.

Syntax:

```
./parent2epn.pl <parenthesized alignment>
```

Typical input (parenthesized) alignment:

```
>secondary structure   “[“ are pseudoknots
(((((-----((((((((-[[-(((((((--------))))--)))--))))))-))-(((((((----)))))))-]])))))
>e. coli
GGUGACGAUAGCGAGAAGGUCACACCCGUUCCCGAACACGGAAGUGAGCUUCUCAGCGCCGACGGUAGAGAGUAGGACGUUGCC
>b. subtillis
CGGCGGCCAUAGCGGCAGGGAAACGCCCGGUCCCGAACCCGGAAGCUAGCCUGCCAGCGCCGAUGGUGGAGAGUAGGUCACCGC
>h. sapiens
GGUGGCGAUAGCAGAGAGGUCACACCCGUUCCCGAACACGGAAGUUAGUUCUCUAGCGCCGAUGGUAGAGAGUAGGACGUUGCC
>s. cerevisiae
GUGGUUAA-AGAAAAGAGGAAACACCUGUUAUCGAACACAGAAGUUAGCUCUUAUUCGCUGAUGGUAGAGAGUAGG-UUAUUGC
>t. pallidum
GUUGCCAU-GGUGGAGAGGUCAUACCCGUUCCCGAACACGGAAGUCAGCUCUCCUACGCCGAUGAUAGAAAGUAGG-UAGUAGC
```

readerpin.pl

This Perl script reads, filters and prints out Erpin outputs. It removes all solutions with 3 undefined nucleotides or more, or with a score lower than a specified threshold.

Syntax:

```
./readerpin.pl <erpin output file> [-fasta] [-c <cutoff>] [-e <cutoff>]

-fasta       : prints sequences in fasta format
-c <cutoff> : prints only sequences with score higher than <cutoff>.
-e <cutoff> : prints only sequences with E-value lower than <cutoff>.
```

tview

This programs pretty prints training set files (.epn), displaying all sequences and the list of structural elements.

Syntax:

```
tview <trset> [<region>]
```

Region: optionally restricts display to a specified region.

Region is expressed as in erpin (e.g. -2,2).

tstat

This programs evaluates the score of any region or mask in the training set sequences (.epn file).

For each region or mask in the command line, `tstat` prints the best/worst/mean score in the alignment, as well as the score cutoffs that should be used to retain 100%, 90%, etc. of the training set sequences. These values can then be used in conjunction with the `-cutoff` argument in Erpin.

Tstat also computes the number of configurations and memory requirement for a given region (useful when dealing with large motifs).

Syntax:

```
tstat <trset> <region> [(-mask | -umask) <arg1> <arg2> ..][..]
```

"Mask" and "region" arguments are the same as for Erpin. For example:

```
tstat trna.epn  -7,7  -umask 7 -umask 8
tstat trna.epn  -7,7  -mask 8
```

5. VERSION HISTORY

5.5	E-values for gapped single strands are now computed based on an approximated formula. Lengthy simulations are not required anymore. The –sss parameter is eliminated. See functions 'GStHist' & 'GetGStHist' in 'libsrc/hshisto.c'
5.4	Bug fix in recursive function 'RecMaskSearch' that affected gap tables. See functions SaveGapsList' & 'RestoreGapsList' in file '/libsrc/mgapslist.c'.
5.3	Extrapolation of score cutoffs >100%. Bug fix: dynamic allocation failed in certain circumstances
5.2.1	Bug fix: problem with single-element motifs
5.1	Bug fixes. Segmentation faults with certain masks. Default substitution matrix for pseudocount is program-encoded. Possible pseudo count values range from 0 to 500
5.0	Improved memory management for large motifs. Very large RNA molecules such as 16S/23S rRNA can be handled in principle.
4.2.5	A significance performance improvement is achieved through optimization of element score assignment. Overall program speed is doubled! A error in pseudocount calculation is corrected. Does not affect results significantly, but default pcw values should now be around 2x10e-4 instead of 10e-1. A corrective factor is applied for backward compatibility. default pcw value is still 10e-1.
4.2.2	Improved E-value (convolution product), Henikoff-like pseudo-counts (-pcw parameter), discontinued Markov representation of gapless single-strands
4.0	stable version w/ E-value
3.9.9.last	fixed bugs
3.9.8.bis	fixed bugs
3.9.5	fixed bugs
3.9.4	fixed bugs
3.9.3	fixed bugs
3.9.2	E-value computing
3.2	Multiple alerts added for excessive CPU and memory usage
3.1.2	secured dynamic mask handler

3.1	code cleaned
3.0.3	fixed bug with very large alignments (16S)
3.0.2	fixed bug
3.0.1	Revised output and parameter formats as in Erpin 2.7
3.0	Introduction of dynamic mask handling. -add parameter introduced
2.9	code cleaned. fixed bug with very large alignments (16S). aug 2002
2.7	Masking must be specified: does not anymore default to -nomask Lowercase replace Asterisks when displaying masked regions. jun 2002
2.5	Fixed bug in precomputed alignment tables (affected multi-level searches)
2.3	Fixed bug in boundary transmission during multi-level search
2.1.1	Fixed bug in search boundaries
2.1	Fixed bug in ungapped single strand scores
2.0	Handling of complex patterns and large alignments Changes in command line and training set format First documented version
1.2	nov 5 2001. a few bugs fixed
1.1	Introduction of "Helix" element
1.0	JMB paper version