MACRO-PERFECTOS-APE — MAtrix CompaRisOn & PrEdicting Regulatory Functional Effect of SNPs

by Approximate P-value Estimation

– User Manual –

October 10, 2014

1 Abstract

Here we present MACRO-APE and PERFECTOS-APE software designed for practical sequence analysis involving classic mononucleotide and dinucleotide position weight matrices (PWMs) of DNA sequence patterns often called *motifs*. The common usage case for DNA motifs is representation of transcription factor binding sites.

The software allows (1) comparing different PWMs using a variant of Jaccard similarity measure, e.g. scanning a motif collection for motifs similar to a given query, (2) analysing single-nucleotide variants for possible regulatory effect through transcription factor affinity changes, (3) performing basic PWM analysis (P-value and threshold estimation).

2 Technical notes

MACRO- and PERFECTOS-APE require Java Runtime Environment 1.6 (or newer) to run. Thus *-APE should be able to function under most modern operating systems.

Several existing motif collections such as HOOCOMOCO as well as several individual PWM examples are available to be used with the *-APE package: HOCOMOCO [http://autosome.ru/HOCOMOCO/] TFBS model collection and several examples of PWMs (motifs) can be downloaded with MACRO-PERFECTOS-APE at [http://opera.autosome.ru/downloads/all_collections_pack.tar.gz].

Windows users can get the latest Java directly from Oracle: [http://www.java.com]. Modern Linux distributions typically have OpenJDK preinstalled, otherwise it should be available via a distribution-specific package manager.

The latest MACRO-PERFECTOS-APE package can be found at [http://opera.autosome.ru/]. Source codes are distributed under WTFPL public license. They are available in a github repository: [https://github.com/prijutme4ty/macro-perfectos-ape] and as a single archive at [http://opera.autosome.ru/downloads/macro-perfectos-ape_src.jar].

This manual is also hosted on github in a repository: [https://github.com/prijutme4ty/macro-perfectos-ape-manual].

3 Overview

All tools are packed in a jar-file with compiled Java classes. There are three main packages for tools: `ru.autosome.ape`, `ru.autosome.macroape` and `ru.autosome.perfectosape`.

APE in `ru.autosome.ape` stands for Approximate P-value Estimation, this package contains basic tools:

- `FindThreshold` — to estimate a PWM score threshold for a given P-value
- `FindPvalue` — to estimate a PWM P-value corresponding to a given score threshold
- `PrecalculateThresholds` — to precalculate lists of thresholds tabulated by P-values for a given motif collection

MACRO-APE in `ru.autosome.macroape` denotes MAtrix CompaRisOn by Approximate P-value Estimation. Package consists of several tools related to motif comparison:

- `EvalSimilarity` — to evaluate similarity for a given pair of PWMs.
- `ScanCollection` — to search a collection of motifs for PWMs similar to a given query.

PERFECTOS-APE in `ru.autosome.perfectosape` denotes Predicting Regulatory Functional Effect of SNPs by Approximate P-value Estimation. Package contains a single tool:

- `SNPScan` — to search a pack of sequences with SNVs or SNPs against a collection of PWMs for (SNV, PWM) pairs, such that single nucleotide substitution induces significant change of predicted affinity for a given PWM.

Please note, that *-APE tools by default consider all given matrices as positional weight matrices with additive scores already passed counts-to-weights transformation (e.g. log-odds). The usage of count matrices (PCMs) or frequency matrices (PPMs) is also possible with additional command-line keys (see the respective sections).

3.1 Command line format

All tools use similar command-line format. The examples are shown under the assumtion that the *-APE package `ape.jar` is located in the current folder (working directory). A typical command line will look like:

```
java -cp ape.jar ru.autosome.ToolName
                 <required arguments>
                 [options]
```

Each tool can be used with `--help` or `-h` options to display a detailed help message describing order of arguments and a list of optional parameters.

Each tool is provided in mononucleotide and dinucleotide versions for mono- and diPWMs and respective background models. Generally, mononucleotide version has wider application range, since most of existing motif collections provide only basic mononucleotide PWMs. Naming convention is the same for all tools: mononucleotide version is located in package's root, dinucleotide version has the same name but is located in a subpackage `".di"`.

E.g. for `ape.FindThreshold` the full class names are:

- `ru.autosome.ape.FindThreshold` for mononucleotide version
- `ru.autosome.ape.di.FindThreshold` for dinucleotide version.

Please note, that dinucleotide tools use special input formats for dinucleotide Position Weight Matrices (diPWM) and respective background models. Input data formats are described in a special section.

3.1.1 Output formats

All tools except `PrecalculateThresholds` print their results into the standard output stream (stdout). `PrecalculateThresholds` stores its results in a set of output files created in a specified folder.

For each tool the output can be redirected to a file using OS syntax, e.g. with a ">"-sign. For example:

```
java -cp ape.jar ru.autosome.ape.FindPvalue motifs/KLF4_f2.pwm
3.3 5.0 7.1 > KLF4_P-values.txt
```

Output generally consists of two types of lines. Lines starting with `"#"` character (comments) show input parameters and descriptions. The results are presented in non-commented lines.

4 Basic APE tools

APE tools are designed to properly convert PWM thresholds to P-values and vice versa.

Position weight matrix (PWM) of DNA motifs assigns a score to each "word" (nucleotide sequence of a fixed length l). It makes possible to range the words by their scores, e.g. corresponding to predicted transcription affinity for PWMs of transcription factor binding sites (TFBS). Given a threshold, one can divide all *l*-mers into two subsets: words whose score are not less than the threshold and the rest. Typically, the words passing the score threshold are selected for downstream analysis, e.g. they are considered as putative transcription factor binding sites.

What is important, the threshold values are not directly comparable for different PWMs. One strategy to have a unified scale is to use motif P-values instead.

The P-value of a certain PWM and a score threshold is the probability to generate a word with the score not less than the threshold at random.

Inverse task is to estimate a threshold for a predefined P-value. In particular this allows to select a PWM score threshold corresponding to a predefined positive prediction rate across the *l*-mer dictionary (e.g. only x% of words are predicted as putative TFBS).

Our tools perform threshold – P-value conversion implementing a dynamic programming algorithm on a granulated (discretized) PWM models using a simplified approach comparing to that described in Touzet et al. [2007].

More details on P-values, thresholds and the algorithm are provided in the MACRO-APE paper. Vorontsov et al. [2013] [`http://www.almob.org/content/8/1/23`]

4.1 FindThreshold

This is a stand-alone tool to search for a score threshold corresponding to a given P-value for a given PWM. `FindThreshold` requires a PWM and a P-value as input and returns a threshold for which the set of words scoring with this PWM no less than the given threshold has the aggregated probability equal to the given P-value. The program can process a set of P-values, and return a set of thresholds. This tool implements a simplified algorithm derived from that implemented in the TFM-Pvalue software of Helen Touzet [`http://bioinfo.lifl.fr/TFM/TFMpvalue/`] but with the fixed predefined discretization level (see section 9.1).

Usage:

```
    java -cp ape.jar ru.autosome.ape.FindThreshold <motif file>
[list of P-values]
```

Example (motif file `KLF4_f2.pat`, P-value of 0.001 and 0.0005):

```
    java -cp ape.jar ru.autosome.ape.FindThreshold motifs/KLF4_f2.pat
0.001 0.0005
```

NOTE! By default `FindThreshold` looks for threshold large enough to obtain P-value not greater than requested (lower boundary for P-value). For details see `--boundary` option description in section 8.4.

4.2 FindPvalue

`FindPvalue` is a stand-alone tool to find the P-value corresponding to a given threshold level for a given PWM.
Usage:

```
    java -cp ape.jar ru.autosome.ape.FindPvalue <motif file>
<list of thresholds>
```

Example (motif file `KLF4_f2.pat`, thresholds of 4.1719 and 5.2403):

```
    java -cp ape.jar ru.autosome.ape.FindPvalue motifs/KLF4_f2.pat 4.1719
5.2403
```

4.3 PrecalculateThresholds

This tool is intended to process the motif collection (a folder containing separate files for each motif) and to store precomputed score distributions of motif PWMs. Each score distributions is saved as a sorted list of (threshold,P-value) pairs with P-values taken at uniform intervals at quantiles of score distribution. It allows for faster score – P-value conversion performing binary search through a list of thresholds or P-values. `PrecalculateThresholds` doesn't store precise score distribution because for a non-disretized PWM it can be extremely large with unpractical precision. Practically it's sufficient to estimate P-value with a specified error level of e.g. 5%.

In order to use precalculated distribution several *-APE tools have `--precalc` option which takes a folder containing results of `PrecalculateThresholds`.

Note: Precalculation allows notably increase speed of threshold to P-value calculation (up to 100x). Unfortunately it deals with a file system to load the precalculated data. Thus it's recomended to use precalculated score distribution for tasks where the same motif P-value evaluation is performed multiple times so that the score distribution is loaded once and used multiple times. At a moment the only use case is – `perfectosape.SNPScan` which assesses each of multiple SNPs against the same motif collection.

Note: Resulting score distribution depends on a discretization level and on a specified background model. It is up to the user to control that a score distribution was precomputed with the proper parameters. The parameters values are not anyhow stored after score distribution precalculation and are not implicitly contolled when reusing precomputed data.
Usage:

```
    java -cp ape.jar ru.autosome.ape.PrecalculateThresholds <motif
collection folder>
<output folder>
[options]
```

Example:

```
    java -cp ape.jar ru.autosome.ape.PrecalculateThresholds ./motifs/
./motif_thresholds/
```

This will create `./motif_thresholds/` folder (if not already exist) and multiple files inside, one file per motif in `./motifs/` folder. For a given motif output file will be named as `<name of motif>.thr`. Each file contains lines in the following format:

`<threshold>` *tab* `<corresponding P-value>`

Lines are sorted with thresholds ascending (P-value descending).

It takes about half a minute to preprocess the collection of ∼400 mononucleotide PWMs with default parameters using 1.5 GHz CPU. During precalculation task progress will be printed to standard error stream. To suppress output use `--silent` option.

To alter granularity of resulting P-values list one can use `--pvalues` option in the following format:

```
--pvalues <from,to,step,mode>
```

Parameters set the P-values progression in the resulting list. P-values can use arithmetic or geometric progession which corresponds to `add` or `mul` value of `mode`.

`from` and `to` represent progression boundaries and `step` corresponds to a common difference (`add`) or a common ratio (`mul`) of progression. Parameters are comma-separated without spaces between.

For example, default progression can be written as follows:

```
--pvalues 1.0,1e-15,1.05,mul
```

It means that `PrecalculateThresholds` collect thresholds for each of these P-values: $1.0, 1.0/1.05, 1.0/1.05^2, 1.0/1.05^3, \ldots, 10^{-15}$

To specify relative error of ϵ use geometric progression with common ratio of $1 + \epsilon$ and boundaries: from 1.0 to a minimal expected non-zero P-value.

5 MACRO-APE: Matrix Comparison by Approximate P-value Estimation

Let us have two PWMs with given threshold levels. The similarity between PWMs is related to the number of words recognized by both PWMs (or the aggregated probability of the word set under the given i.i.d. model). To calculate this value we use generalized approach described in Touzet et al. [2007] for two PWMs simultaneously in a way similar to that in Pape et al. [2008]. The number of words recognized by both PWMs can be used to construct a variant of Jaccard similarity measure for motifs considered as sets of allowed words scoring no less than predefined thresholds.

Typical methods of PWM comparison are based on direct evaluation of matrix elements, for instance by comparing matrices column by column (where different columns correspond to different positions of a transcription factor binding site). On the other hand, in applications PWM is used as TFBS model to identify binding sites by scanning a given sequence and identifying words with scores no less than a threshold.

Thus, in reality a TFBS model is related to the set of words scoring no less than the given threshold for the given PWM. It is desirable to construct a similarity measure for TFBS models based on the similarity between word sets recognized by the matrices with given thresholds, rather than on similarity between matrices per se. Moreover, comparison-by-elements strategy requires the matrices to have algebraically comparable values (either frequencies or specifically scaled weights) which is not necessary if sets of recognized TFBS are compared.

MACRO-APE computes a similarity measure which directly accounts for similarity of recognized word sets. This measure does not require PWM elements to be algebraically comparable and so it can be used to compare weight matrices constructed by different normalization / conversion strategies (e.g. log-odds with different pseudocounts and/or background normalization).

Let us have a position weight matrix of length l. The whole set of ACGT-alphabet words of length l will be called the dictionary of size $N = 4^l$. For a fixed threshold level t one can calculate the fraction of the dictionary (i.e. the number of words n) scoring no less than the threshold. We will call the value of n/N as the motif P-value.

Suppose we have two PWMs m_1, m_2 of length l and some P-value levels p_1, p_2. For m_1 and m_2 we can estimate the thresholds t_1 and t_2 corresponding to p_1, p_2. Having PWMs with the corresponding thresholds we can estimate the fraction f of the dictionary recognized by both models, i.e. the size of the set of words scoring no less than t_1 on m_1 and no less than t_2 on m_2.

Moreover one can construct the Jaccard index

$$J = \frac{\mathbf{A} \cap \mathbf{B}}{\mathbf{A} \cup \mathbf{B}}\,, \tag{1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are sets of words recognized by m_1 and m_2 with the thresholds t_1 and t_2. If necessary one also can construct a Jaccard distance as

$$d(\mathbf{A}, \mathbf{B}) = 1 - J\,. \tag{2}$$

In the general case we have two PWMs of different widths, unknown optimal mutual alignment and orientation. For each possible alignment shift and orientation the matrices can be extended to the same length by adding zero-columns (not affecting either score or threshold) and then compared as the two models of the same width. Then one can determine the optimal shift and orientation by selecting the case with the highest Jaccard similarity. More formal and detailed explanation can be found in the corresponding macroape paper Vorontsov et al. [2013].

NOTE! The reverse complementary transformation can be necessary to optimally align a given pair of matrices, thus the background nucleotide composition for matrix comparison tools should be symmetrical, i.e. p(A) = p(T) and p(C) = p(G).

5.1 EvalSimilarity

`EvalSimilarity` computes the similarity of two given motifs defined as a Jaccard similarity of sets of words recognized by each motif. Optimal mutual alignment of the motifs is also estimated. Sets of recognized words are given by a PWM accompanied with threshold or a P-value.

By default a set of recognized words is defined as top 0.05% of words (i.e. P-value level of 0.0005) ranked by a PWM. It's possible to set required P-value with `--pvalue <P-value>` option or to specify thresholds explicitly so that word sets contain all words passing corresponding thresholds. It can be accomplished using `--first-threshold <threshold>` and `--second-threshold <threshold>`.

In order to get intuition of Jaccard similarity scale and to better catch our output format, try these examples and take a look at corresponding motif logos (see the sample data):

Example (rather similar motifs `KLF4_f2` and `SP1_f1`, see fig. 1):

```
    java -cp ape.jar ru.autosome.macroape.EvalSimilarity motifs/KLF4_f2.pat
motifs/SP1_f1.pat
```

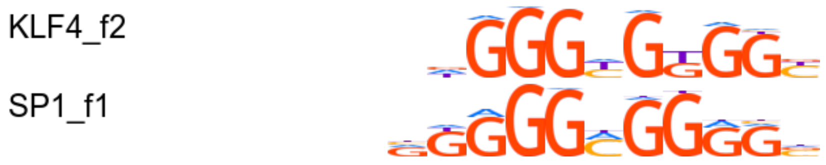

Figure 1: Sequence logo corresponding to a motif alignment.

Example (the same motif `SP1_f1` in opposite orientations):

```
    java -cp ape.jar ru.autosome.macroape.EvalSimilarity motifs/SP1_f1_revcomp.pat
motifs/SP1_f1.pat
```

Example (significantly different motifs `SP1_f1` and `GABPA_f1`):

```
    java -cp ape.jar ru.autosome.macroape.EvalSimilarity motifs/SP1_f1.pat
motifs/GABPA_f1.pat
```

By default `EvalSimilarity` tests all possible mutual motif alignments in both orientations. A special option `--position` will force evaluating similarity with the explicitly specified motif alignment:

`--position <shift>,<direct|revcomp>`

Option parameters are comma-separated, spaces not allowed; the position is defined for the second motif relative to the first.

Try the following examples:

Example (rather similar motifs `KLF4_f2` and `SP1_f1` at optimal alignment):

```
    java -cp ape.jar ru.autosome.macroape.EvalSimilarity motifs/KLF4_f2.pat
```

```
motifs/SP1_f1.pat
--position -1,direct
```

Example (rather similar motifs `KLF4_f2` and `SP1_f1` at completely wrong alignment):

```
java -cp ape.jar ru.autosome.macroape.EvalSimilarity motifs/KLF4_f2.pat
motifs/SP1_f1.pat
--position 3,revcomp
```

Note! By default `EvalSimilarity` selects the thresholds corresponding to the P-value not less than requested (upper boundary) possibly making compared word sets larger (not to miss words with scores too close to the threshold). This differs from `FindThreshold` approach which, by default, uses lower boundary for P-valuethus controlling the prediction rate more strictly.

It is very important to select upper P-value boundary for short PWMs. In case of given low P-values they can recognize no words at all (so the Jaccard measure may have zero numerator and zero denominator). For reasonable threshold levels both upper and lower boundaries usually produce very close similarity values, see the MACRO-APE paper for details Vorontsov et al. [2013].

Nevertheless, one can override this behavior with `--boundary lower` option. In such a case if any of supplied PWMs recognizes no words for a selected P-value, then similarity can not be correctly determined and macroape will report the similarity value of -1.

5.1.1 ScanCollection

This tool uses a collection of motifs to find PWMs similar to a given query. The list of similar PWMs is sorted by similarity in descending order so the PWMs similar to the query are located at the top of the list.

NOTE! The shift and orientation are reported for PWMs from the collection relative to the query PWM.

Example(search for motifs similar to `KLF4_f2`, HOCOMOCO collection):

```
java -cp ape.jar ru.autosome.macroape.ScanCollection motifs/KLF4_f2.pat
./hocomoco/
```

The two-pass search mode is available to recheck the top part of the list using a more precise discretization level. Second pass is executed only if `--precise [min_similarity=0.01]` key is specified. The precise search will recheck only the PWMs similar to the query with a similarity no less than `min_similarity`. The results of the second pass will be marked by asterisk(*).

One can specify similarity cutoff with option `--similarity-cutoff <similarity cutoff>` or `-c <similarity cutoff>` to discard comparison results with the resulting similarity less than a given value (the 1st pass results are used). By default, records with similarity less than 0.05 are not shown. In order to print comparison results for all PWMs in collection `--all` option can be used.

Example(search PWMs similar to `KLF4_f2`, extended precision for the most similar PWMs):

```
    java -cp ape.jar ru.autosome.macroape.ScanCollection motifs/KLF4_f2.pat
./jaspar/
--precise
```

To find similar PWMs using a particular P-value level one should use the `"--pvalue"` option. Default P-value is 0.0005.

Example ():

```
    java -cp ape.jar ru.autosome.macroape.ScanCollection motifs/KLF4_f2.pat
./selex/
--pvalue 0.001
--similarity-cutoff 0.06 --precise 0.1
```

6 PERFECTOS-APE: Predicting Regulatory Functional Effect of SNPs by Approximate P-value Estimation.

Variations in genome sequences are quite common. One widespread type of variations is represented by single nucleotide substitutions called single nucleotide variants (SNVs) or, for a given population, single nucleotide polymorphisms (SNPs).

SNVs in gene regulatory regions may affect gene expression through alterations in transcription factor binding sites.

PWM of transcription factor binding sites provides a score for any putative TFBS. This score roughly represents binding affinity, thus allowing to estimate the impact of a given substitution through change in a score value.

As discussed earlier (section 4) scores are not directly comparable and do not have a unified scale. More convenient measure is the P-value - the probability to find a high-scoring word at random.

PERFECTOS-APE computes motif P-values for each sequence variant and calculates P-value fold change of a given substitution. Detailed algorithm for evaluating a fold change for a given TF and a substituion:

- Calculate PWM scores for putative TFBS overlapping a sequence variant.
- Choose the best position and score for both sequence variants independently.
- Estimate P-values for the best scores.
- Compute fold change as the rate of P-values.

PERFECTOS-APE tests given SNVs against a whole collection of PWMs and yields (SNV, TF) pairs of SNVs that may significantly affect TF affinity.

6.1 SNPScan

`SNPScan` takes a list of SNVs with flanking sequences and a motif collection and returns a list of predicted TFBS which were possibly disrupted by or emerged after a certain SNV.

Usage:

```
    java -cp ape.jar ru.autosome.perfectosape.SNPScan <path to the
collection of motifs> <path to the file with the list of SNVs> [options]
```

`SNPScan` has two filters. The first discards (SNV, TF) pairs without TFBS prediction at any of nucleotide variants. `SNPScan` treat a word as a putative TFBS if P-value of this word's score is not greater than the predefined threshold (0.0005 by default, changed via `--pvalue-cutoff` option:
`--pvalue-cutoff <maximal P-value to be considered>`
or in short form:
`-P <maximal P-value to be considered>`.

The second filter requires check P-value fold change to be large enough. By default fold change threshold is equal to 5. It means that only SNVs causing P-value change of 5x and more (FoldChange¿5 or FoldChange¡1/5) will be included in results. Fold change threshold can be specified using `--fold-change-cutoff`:
`--fold-change-cutoff <minimal fold change to be considered>`
or in short form:
`-F <minimal fold change to be considered>`

The last but the most useful option is `--precalc` which forces `SNPScan` to work with precalculated P-value,thresholds pairs performing binary search to evaluate the P-value instead of calculating motif score distribution each time from scratch. It can reduce total computation time in hundreds of times for large datasets. As an input it requires a folder with precalculated (P-value,threshold) pairs - one for each motif:
`--precalc <path to a folder with precalculated P-value, threshold pairs>`

These precalculated score distributions are to be produced by a `PreprocessCollection` from APE toolbox. Please refer to the respective section for details.

Example:

```
    java -cp ape.jar ru.autosome.perfectosape.SNPScan ./hocomoco/pwms/
snp.txt --precalc ./collection_thresholds
    java -cp ape.jar ru.autosome.perfectosape.SNPScan ./hocomoco/pcms/
snp.txt --pcm --discretization 10 --background 0.2,0.3,0.3,0.2
```

6.1.1 Output data format

`SNPScan` prints all results to standard output, errors and messages go into standard error stream. First line of output is a header of table. Latter lines are rows of this table. Columns are:

- Name of sequence containing SNV

- TF motif name
- for the first allele variant:
  - the best position and strand of putative TF-DNA binding
  - nucleotide word corresponding to the best binding sequence among all other words in sequence, intersecting SNV
- the same two columns for the second allele variant
- allele variants
- P-value for the first allele variant
- P-value for the second allele variant
- fold change (the first P-value divided by the second P-value)

Position of the best binding place is given for the leftmost boundary of a binding sequence (independent of strand orientation). The SNV location is at zero, so the TFBS coordinates are always less or equal to zero. Strand is denoted as ‘direct‘ or ‘revcomp‘. Words are given at the relevant strand (i.e. reverse-complement transformation is applied if necessary).

7 Data formats

7.1 Position matrix format description

All tools in the *-APE package use the following matrix file format (each binding site position corresponds to a separate line):

```
some_header
pos1_A_weight  pos1_C_weight  pos1_G_weight  pos1_T_weight
...
posw_A_weight  posw_C_weight  posw_G_weight  posw_T_weight
```

Position matrix format is appliable for all kinds of positional matrices: positional weight(PWM), count(PCM) and probability/frequency(PPM). Positonal count matrices are allowed to contain floating point numbers (e.g. in the case the counts were derived from somehow weighted alignments).

The total number of lines corresponds to the PWM width (minus the header line). If given, header will be treated as a motif name, otherwise filename will stand for motif name. Header may carry an optional ">" sign at line start (like in fasta files).

If necessary it’s possible to read transposed matrices, with nucleotides in rows and positions in columns using `--transpose` option.

Example (PWM similar to HOCOMOCO transcription factor motif for KLF4):

```
>KLF4_f2
0.308 -2.254 0.135 0.328
-1.227 -4.814 1.305 -4.908
-2.443 -4.648 1.358 -4.441
-2.717 -3.807 1.356 -3.504
-0.556 0.534 -3.614 0.527
-1.868 -4.381 1.337 -3.815
-2.045 -2.384 0.719 0.544
-1.373 -3.006 1.285 -2.502
-2.103 -1.894 1.249 -1.428
-1.327 0.898 -0.808 -0.181
```

Example (Transposed PWM similar to HOCOMOCO transcription factor motif for KLF4):

```
> KLF4_f2
1233.5  264.0   76.8    57.9    518.2   138.0   115.3   227.8   108.7   238.5
93.2    5.3     6.6     18.1    1545.9  9.3     81.5    42.8    134.5   2226.0
1036.6  3347.6  3529.5  3520.3  22.4    3456.3  1861.9  3278.6  3162.9  402.4
1258.3  4.7     8.6     25.2    1535.0  17.9    1562.8  72.2    215.4   754.7
```

More real-life examples are provided with the package in respective motif collections.

Dinucleotide versions of *-APE tools use dinucleotide motifs. Dinucleotide positional matrices have similar format but contain 16 columns instead of 4. Columns go in order: AA, AC, AG, AT, CA, CC, . . . , TT. It's also possible to use mononucleotide motifs in dinucleotide tools (e.g. to use dinucleotide background). For rationales and details take a look at `--from-mono` option.

7.2 SNVs/SNPs format

`SNPScan` uses a list of sequences with SNVs as input data.

The list of sequences with SNVs should be given in a single plain text file. Each sequence should be presented at a separate line using the following format:
`<SNV name> <left flank>[<variant 1>/<variant 2>]<right flank>`

SNV name shouldn't contain empty delimiters (spaces or tabs). Sequence consists of two allele variants in square brackets, separated with '/', and flanking sequences at both sides. Length of flanking sequences should be sufficient to place the longest motif of a given collection (so it is advised to provide 25-30bp at each side) into all positions relative to a nucleotide substitution position.

Example (SNV list):

```
rs10040172 gatttgccctgattgcagttactga[G/A]tggtacagacatcgtaataatctta
rs10116271 taaattctatgtggggaagaggtct[C/T]gtagaggcgatgattcttacattgc
rs10208293 cttcatacatttatgtccagtacct[A/G]tggaccctccttgtgaactcttctc
rs10431961 tggcggggctggtcaggcggcgtcg[C/T]cggtacgctctgagcggcagcgtgt
```

8 Additional command-line options

Many additional options are available for *-APE tools. The options should be provided after the required arguments. There are common options among all *-APE tools as well as tool-specific options (already described in the respective sections).

This section covers common options: those altering input data format and those affecting calculation parameters. The first class of options allows using input motifs as different matrices (counts, PCM or probabilities, PPM) instead of default weights (PWM), load matrices in transposed format and use mononucleotide motifs in dinucleotide tools. The second class of options allows to set the background model, select P-value evaluation mode, limit memory consumption and so on.

For a full list of options for a particular tool please run the tool with the `--help` command line option.

8.1 Option families

Options are grouped into "families" of options with similar names but different prefixes. For example `macroape.di.EvalSimilarity` tool, has an option `--from-mono`. This option creates dinucleotide motifs by loading mononucleotide matrices. In turn, `--first-from-mono` options forces loading of the first motif from mononucleotide input and `--second-from-mono` does the same for the second motif.

Similar options for `macroape.ScanCollection` are named `--query-from-mono` and `--collection-from-mono`. Option `--query-from-mono` requires mononucleotide query matrix, and `--collection-from-mono` means that each motif in collection should be loaded from mononucleotide matrix. The same is appliable for `--background`.

Such triples of options are typically listed in the help string like this: `--[first-|second-]from-mono`. It means that one can use both prefixed and non-prefixed options. Possible prefixes are given in square brackets separated with a pipe sign `"|"`.

Note: Prefixed options exist only in a long form. E.g. one can use both `-b` and `--background` as synonymous but for there is no short analogue for `--first-background`.

Note: Presence of separate options for each of used motifs doesn't necessarily involve existence of a common option. E.g. `macroape.EvalSimilarity` has `--first-threshold` and `--second-threshold` options but doesn't have `--threshold` since it generally makes no sense to use the same algebraical threshold value for two independent PWMs (common P-value level in turn is a reasonable parameter).

8.2 Motif loading options

By default motifs are expected to be provided as position weight matrices in a nucleotides-in-columns plain text format. Basic tools use mononucleotide positional matrices, dinucleotide tools use dinucleotide matrices. However, many motif collections provide position frequency matrices (PFMs, or probability matrices, PPMs) or position count matrices (PCMs). *-APE tools can convert these matrices to PWMs internally (using a log-odds-like transformation as in Lifanov et al. [2003], see the section 9).

$$\begin{tikzcd}
PPM \arrow[r, "\texttt{--ppm}"] & PWM\bullet \arrow[d, "\texttt{--from-mono}"] & PCM \arrow[l, "\texttt{--pcm}"'] \\
diPPM \arrow[r, "\texttt{--ppm}"] & diPWM\bullet & diPCM \arrow[l, "\texttt{--pcm}"']
\end{tikzcd}$$

Figure 2: Command-line options to read a motif from non-PWM motif models. Conversion end-points are marked with bullets.

$$\begin{tikzcd}
PPM \arrow[r, "\texttt{--effective-count}"] & PCM \arrow[r, "\texttt{--background} \\ \texttt{--pseudocount}"] & PWM\bullet \arrow[d] \\
diPPM \arrow[r, "\texttt{--effective-count}"] & diPCM \arrow[r, "\texttt{--background} \\ \texttt{--pseudocount}"] & diPWM\bullet
\end{tikzcd}$$

Figure 3: Motif transformations configuration options. Conversion end-points are marked with bullets.

8.2.1 Obtaining PWM from PCM and PPM models

To load motif from position count matrices there is a special `--pcm` option. A similar option `--ppm` words for positional probability matrices (see fig. 2).

The PCM → PWM or PPM → PWM data model transformations can be configured.

The PCM → PWM conversion is described in a section 9.2. It's possible to manually specify a fixed pseudocount a with `--pseudocount <a>` option. When not specified, pseudocount is derived from alignment weight W: $a = \ln(W)$.

PPM → PWM conversion is done in two stages. At first PPM is multiplied by a constant alignment weight W to obtain a PCM. Then this PCM is converted to a PWM as described above. For the PPM → PWMconversion, a user should supply alignment weight W (for example it can be the total count of words in the initial alignment) explicitly by the `--effective-count <W>` option. If this information is not given, alignment weight of 100.0 will be used as a default assumption.

PCM → PWM conversion will take the user-specified background into account.

DiPCMs are converted to diPWMs using the same formula as for PCM → PWM conversion, the only difference is that now nucleotide index goes through 16 dinucleotides at each position instead of 4 nucleotides.

Possible configuration options can be seen on a fig. 3.

8.2.2 Obtaining dinucleotide motifs from mononucleotide ones

Dinucleotide *-APE tools take dinucleotide motifs as input parameters. But there is an option `--from-mono` which allows to use basic mononucleotide motifs instead so that PWM $\rightarrow$ diPWM will be done internally. It can be useful in following cases:

- Comparison of dinucleotide motif against mononucleotide one. In this case one motif should be loaded as dinucleotide motif, the rest - as mononucleotide motif internally converted to a dinucleotide motif. Further comparison performs on two dinucleotide motifs.

- Study of mononucleotide motif properties on dinucleotide background. It isn't possible to specify dinucleotide background for a mononucleotide tool, but is possible to specify mononucleotide motif and dinucleotide background for a dinucleotide tool.

PWM $\rightarrow$ diPWM is done in such a way that each word has the same score on diPWM as it had on PWM.

Notice: scores of words on discreted PWM and corresponding diPWM can be slightly different due to a discretization step performed after PWM $\rightarrow$ diPWM conversion. This discrepancy shouldn't worry you, it's small enough and goes to zero with discretization increase.

When both `--pcm` and `--from-mono` options are specified, the conversion is done in two stages (see fig. 3). First, PCM $\rightarrow$ PWM transformation is applied and then PWM $\rightarrow$ diPWM transformation is applied. Background model used in PCM $\rightarrow$ PWM conversion should be given as mononucleotide letter frequencies ("Bernoulli" i.i.d. random model). Background provided to a dinucleotide tool should be given as dinucleotide frequencies. In this case mononucleotide frequencies are estimated by averaging dinucleotide background:

$$p_\alpha = \frac{\sum_\beta p_{\alpha\beta} + \sum_\beta p_{\beta\alpha}}{2} \tag{3}$$

8.2.3 `--transpose` option

One can load motifs from nucleotides-in-rows using `--transpose` option. The only difference in format is matrix orientation, header remains the same (see section 7.1).

8.3 Background model options

Nucleotide frequencies of a background model can be specified in optional arguments, e.g. `--background` or `--query-background`. All background options use the same format with a single required argument: `--background <value>`.

Default background model is a `wordwise` model. It means that all our calculations assume uniform nucleotide distribution and the exact number of words is used everywhere instead of probabilities of a word set. E.g. `FindPvalue` will calculate not the probability of a random word score to pass the threshold but a fraction of words scoring greater than threshold estimating the exact number of such words.

A number of words is a more natural and intuitive to use, especially if the background model cannot be properly selected thus we suggest ”wordwise” mode by default.

Wordwise mode can be specified explicitly, e.g. using `--background wordwise` key.

All following formats are different ways to specify frequencies of each nucleotide:

- The most simple nucleotide background model is uniform, each nucleotide has the same probability to occur. Option format is: `--background uniform`. This is close to wordwise mode, but word set probabilities are used and reported instead of raw counts of words.

- It is also possible to specify a fixed GC-content (in range 0 to 1): `--background <GC-content>`. E.g. `"--background 0.6"`

- The most detailed format is to explicitly specify nucleotide frequencies: `--background` $<p_A, p_C, p_G, p_T>$. E.g. `"--background 0.2,0.3,0.3,0.2"` will define the same frequencies as for GC-content of 0.6. Note that nucleotide frequencies should be given in alphabetical ACGT-order separated with commas.

Note: No spaces between frequencies are allowed (commas only). Sum of frequencies should be equal to 1.0.

8.3.1 Dinucleotide background options

Dinucleotide background for dinucleotide tools has the same variants: wordwise, uniform, GC-content and full list of dinucleotide frequencies. Wordwise, uniform and GC-content backgrounds are effectively the same as mononucleotide ones and don't carry any nucleotide interdependencies.

Dinucleotide frequencies require additional clarification. Dinucleotide frequencies should be given in an alphabetical order: AA, AC, AG, . . . , TT — 16 terms.

Each value corresponds to a probability of a specific dinucleotide. These probabilities are **not** conditional probabilities used by an algorithm internally, but actual dinucleotide frequencies. Please be careful if you already got used to use Markov model background.

Again, list of probabilities is comma-separated, no spaces allowed, sum of probabilities should be equal to 1.0.

Also one can specify mononucleotide ACGT-frequencies background for dinucleotide tools. It will be recognized automatically when 4 values are specified instead of 16.

8.4 Additional command-line options

8.4.1 Specifying custom discretization level

For a more precise result `--discretization <discretization rate>` or `-d <discretization rate>` command line key can be used to explicitly set the discretization level for PWM elements, like `"--discretization 100000"` (see the section 9.1). The discretization level of 10^5 corresponds to the precision of PWM elements up to 5 decimal places. A larger number of decimal places results in increased precision and computational time. The default setting of 10^4 for single-motif tools and 10^1 for motif comparison tools gives reasonable "time-precision" tradeoff.

8.4.2 Specifying custom P-value level

All tools in `MACRO-APE` package estimate motif threshold by a P-value for further use. By default P-value level of 0.0005 is assumed. It can be overriden with `--pvalue <P-value>` or `-p <P-value>` option key.

8.4.3 Choose proper threshold by a P-value

All *-APE tools except `ape.FindPvalue` and `perfectosape.SNPScan` perform internal P-value to threshold conversion. Since PWM P-values have discrete distribution a given P-value can be achieved only approximately. A fixed threshold corresponds to the actual P-value which is smaller or larger than the requested P-value.

The boundary selection can be done using `--boundary <lower|upper>`.

For model comparison by default we use the upper boundary for the P-value (so even at low given P-values PWMs recognize some words and thus the models can be compared). If searching for a threshold corresponding to the given P-value we report the lower boundary of the P-value by default (to properly control the positive prediction rate corresponding to a given threshold).

Note: `lower` boundary means that P-value will be not greater than the requested one. The threshold for `lower` P-value will be greater than the threshold for `upper` boundary P-value.

8.4.4 Limiting CPU and memory consumption

It's possible to create an artificially arranged PWM whose score distribution will grow exponentially with length and thus can take a lot of memory and time for computation. This option is mostly designed to prevent *-APE tools from unnormal CPU and memory consumption. If hash size exceeded a given limit, tools cancel calculations with `"Hash overflow"` error message. In such case user can manually expand hash size limits or lower discretization level.

- `--max-hash-size <size>`: set the internal hash (used for score distribution calculation) size limit. Default value is 10^7

- `--max-2d-hash-size <size>`: set the internal two-dimensional hash size limit (used for PWM comparison in `MACRO-APE` toolbox). Default value is 10^4.

9 Formal math

9.1 PWM discretization

Following the general idea described in Touzet et al. [2007] we can effectively calculate the P-value for a given PWM with a fixed precision and a given threshold value. The algorithm of Touzet *et al.* efficiently processes matrices with integer elements. The matrices with real values are transformed into integer value matrices by multiplying each value by discretization constant and truncating the decimals.

Effectively this is similar to rounding up real values leaving only the fixed number of decimal places. The higher discretization level will result in a more accurate P-value calculation and an increased computational time.

Please note, that in contrast to the original Touzet algorithm here we applying "ceil" operation to the matrix elements (instead of "floor" in the original paper of Touzet). This allows us to have a strict upper boundary of the threshold for a given P-value.

We use the default discretization level of 10^4 to perform calculations with accuracy up to four significant digits for single-PWM tools from APE toolbox.

For motif comparison the straightforward discretization by rounding up to the nearest integer is used by default for a fast and rough search through the motif collection. The default level of 10 (one decimal place) is used for a more precise search of similar motifs.

Thus in our case discretization is the transformation as follows: discretized S is S multiplied by discretization level V and rounded up to the nearest integer value.

```
Example:
S = 1.6734
discretization V=1    discretized S = ⌈1.6734⌉ = 2
discretization V=10   discretized S = ⌈16.734⌉ = 17
discretization V=100  discretized S = ⌈167.34⌉ = 168
```

Discretization will generally preserve the word score ranking with the common exception for words that would obtain identical scores. The main advantage of the discretization is decreasing of the number of possible scores so the set of all possible scores can be enumerated more effectively.

9.2 PCM to PWM conversion algorithm

Matrix of positional counts (PCM) can be transformed to PWM using the following formula Lifanov et al. [2003]:

$$PWM_{\alpha,j} = \ln \frac{PCM_{\alpha,j} + aq_\alpha}{(W + a)q_\alpha}\ , \tag{4}$$

where α is a nucleotide (or dinucleotide) index and j is a position index; W is the total weight of the alignment (or the number of aligned words), a is the pseudocount value, and q_α is the background probability of nucleotide letter α.

Pseudocount is taken by default as the $\ln W$ but can be explicitly specified by user.

Alignment weight W is typically a total number of aligned words and can be calculated from a given PCM as a sum of nucleotide counts in a particular column : $W_i = \sum_\alpha PCM_{\alpha,i}$. W_i is the alignment weight for i-th position. Typically each position has the same alignment weight W, but multiple local alignment algorithms may produce positional count matrices with different weights W_i of words covering each position (e.g. flanks can have less weight than a central part of motif). Thus the weight is safer to calculate separately for each motif position.

For PCM → PWM conversion *-APE tools use a slightly modified formula:

$$PWM_{\alpha,j} = \ln \frac{PCM_{\alpha,j} + a_j q_\alpha}{(W_j + a_j) q_\alpha} \tag{5}$$

Here a_j is a pseudocount related to j-th position. It can be either fixed for each position or equal to a logarithm of corresponding alignment weight: $a_j = \ln W_j$

By default all tools accept weight matrices (i.e. already converted using any similar procedure).

References

Alexander P Lifanov, Vsevolod J Makeev, Anna G Nazina, and Dmitri A Papatsenko. Homotypic regulatory clusters in drosophila. *Genome research*, 13 (4):579–588, 2003.

Utz J Pape, Sven Rahmann, and Martin Vingron. Natural similarity measures between position frequency matrices with an application to clustering. *Bioinformatics*, 24(3):350–357, 2008.

Hélène Touzet, Jean-Stéphane Varré, et al. Efficient and accurate p-value computation for position weight matrices. *Algorithms Mol Biol*, 2(1510.1186): 1748–7188, 2007.

Ilya E Vorontsov, Ivan V Kulakovskiy, and Vsevolod Makeev. Jaccard index based similarity measure to compare transcription factor binding site models. *Algorithms for Molecular Biology*, 8(1):23, 2013.