Overview of the *PWMEnrich* package

Robert Stojnić[*]

April 4, 2013

Contents

1 Introduction **1**
1.1 Implemented algorithms 1
1.2 S4 class structure and accessors 3

2 Motif enrichment **3**

3 Parallel execution **6**

4 Using a custom set of PWMs **6**

5 Using a custom set of background sequences **7**

6 Session information **7**

1 Introduction

The purpose of this package is to asses the enrichment of already known Position Weight Matrices (PWMs) in a set of DNA sequences. Note that this is not the same as *de-novo* motif finding which discovers novel motifs, and motif comparison which aligns motifs. The main difference to traditional PWM-based sequence scanning is that PWM hits are considered *together* in a sequence or DNA region of certain length, instead of being given individual P-values for individual motif hits. One example of such an approach which we re-implement in the package is Clover (Frith et al., 2004).

The package is useful both for hypothesis verification, e.g. testing if a ChIP-chip/seq dataset is enriched for a target TF, and for hypothesis generation, e.g. finding novel co-factors (with already existing PWMs) in an already defined regulatory region(s). A variety of algorithms has been implemented to asses enrichment using both motif hit-count based metrics and threshold-free metrics. Enrichments can be normalized to a background (e.g. genomic) distribution and P-values derived.

1.1 Implemented algorithms

Various algorithms are implemented for PWM enrichment analysis: with/without fixed threshold and with/without background correction. The DNA sequence is scanned with a set of Position Weight Matrices (PWMs) corresponding to transcription factor (TF) binding affinities. The goal is to find a set of PWMs that have the largest affinity for the sequence and thus are most likely to functionally bind to the sequence(s) of interest.

[*] e-mail: robert.stojnic@gmail.com, Cambridge Systems Biology Institute, University of Cambridge, UK

At present, the package has been developed and tested in *Drosophila melanogaster*, however, it is more broadly applicable as it contains tools to build backgrounds for custom organisms and scan with a set of custom PWMs. The algorithms are applicable to single sequences, groups of sequences, and comparisons of single and groups of sequences (i.e. differential enrichment).

Every algorithm has two steps. In the first, sequences are scanned to assess the affinity of a PWM to them. In the second, background correction is performed. For the first step the following metrics are available:

- **Fixed-cutoff motif hit** - for each sequence number of significant motif hits over a score cutoff is recorded. This cutoff can be fixed to a log2-odd value, or can be determined by specifying a P-value. The P-value is converted into a fixed threshold by examining its empirical distribution on a large set of background sequences.

- **Threshold-free affinity** - instead of introducing a fixed threshold, this approach takes the average odds (not log-odds!) score over the whole sequence to produce a MeanAffinity score. Theoretical work suggests that this quantity is related to the average time TF spends bound to a DNA sequence (Stormo, 2000). Recent benchmarks suggest this as a method of choice (Stojnic and Adryan, 2012).

PWMs are sorted according to number of significant motif hits or MeanAffinity. However, these numbers might not be an optimal indication of motif enrichment in the sequences. For example, a motif with low information content will have, on average, higher number of hits than a motif with high information content. Thus, it is important to perform background correction, especially when analysing more than one sequence (Stojnic and Adryan, 2012).

The background is typically fitted to a set of all promoters in an organism. The following background correction algorithms are available in the package:

- **Z-score** - applicable to fixed-threshold motif hits. The density of binding sites above the threshold is calculated in the background set of sequences and Z-score calculated by using a normal approximation to the sum of binomial random variables representing motif hits along the sequence. This approach is described in more detail in (Sui et al., 2005).

- **Lognormal** - applicable to the threshold-free MeanAffinity score. The background distribution of MeanAffinity scores is assumed to be distributed according to the lognormal distribution with standard deviation depending on sequence length. The approach compared favourable in benchmark on gold-standard datasets and is recommended in all contexts and is described in (Stojnic and Adryan, 2012).

- **Generalized extreme value (GEV)** - applicable to the threshold-free MeanAffinity score. The background distribution of MeanAffinity scores is assumed to be distributed according to the Generalized extreme value (GEV) distribution. The three parameters of this distribution are fitted with linear regression on a range of typical regulatory region sizes. This approach is described in (Manke et al., 2008).

- **Matrix shuffle** - applicable for both fixed-threshold and threshold-free score. No background set of sequences is used in this approach, instead, PWMs are shuffled by reshuffling PWM columns many times. This creates a null-distribution of motif scores for each of the motifs separately. This approach is very slow since the sequence needs to be rescanned many times. We assume that the distribution of shuffled scores is roughly normal and we calculate the Z-score of observed score. This approach is described in (Bodén and Bailey, 2008).

- **Empirical P-value** - applicable for both fixed-threshold an threshold-free score. The P-value of a score is directly estimated by sampling datasets with same sequence lengths from a set of background sequences. As the P-values typically get smaller with higher number of sequences, the approach is practically applicable only to single sequences and requires the background set that is at least 1/minimal.P.value times larger than the sequence to ensure that the tails of the empirical distribution can be sampled properly. For instance, if one wants to find an empirical P-value for a 500bp sequence with a P-value resolution of 0.001, one would need to have a background set of at least 500bp * 1000 = 500kb and do at least 1000 randomizations (recommended at least 10000 to gain more confidence).

All of these background corrections are applicable to single sequences and groups of sequences when the group is treated as one long sequence. To treat the group of sequences as a group of equal ”regulatory blocks” of which an unknown subset has the specified motif we implement the Clover (Frith et al., 2004) algorithm.

1.2 S4 class structure and accessors

The *PWMEnrich* package builds upon the *Biostrings* package and uses the classes from this package to represent DNA sequences (`DNAString` and `DNAStringSet`). It introduces a new class `PWM` to represent a PWM together with the frequency matrix and other parameters (background nucleotide frequencies and pseudo-counts). All motif scoring is performed by the *Biostrings* package which is why the *PWMEnrich* package also returns log2 scores instead of more common log base e scores.

The package also introduces a number of classes that represent different background distributions: `PWMLognBackground`, `PWMCutoffBackground`, `PWMEmpiricalBackground`, `PWMGEVBackground`. In all cases, the classes are implemented with a list-like (and RefClass) interface, that is, individual pieces of information within the objects are accessibly using `names(obj)` and `obj$prop`.

2 Motif enrichment

For *D. melanogaster* and the Bioconductor MotifDb motif database the package provides a set of already compiled background correction objects. See the documentation for MotifDb for more information on sources for these motifs. Sections 4 and 5 describe how to create custom background correction objects.

We will load two DNA sequences from a FASTA file and then perform motif enrichment of the 740 *D. melanogaster* PWMs using lognormal background correction. The first sequence is known to bind Tin, and the second is known to bind Twi.

```
> library(PWMEnrich)
> library(PWMEnrich.Dmelanogaster.background)
> # load the pre-compiled lognormal background
> data(PWMLogn.dm3.MotifDb.Dmel)
> # scan two sequences from a FASTA file for motif enrichment
> sequences = readDNAStringSet(system.file(package="PWMEnrich",
+   dir="extdata", file="example.fa"))
> sequences

  A DNAStringSet instance of length 2
    width seq                                               names
[1]   350 CATGTCAAGTGGCACTAAACATG...GTCGCAGCTGCGAGCCTCCCACC tinD
[2]   183 GTCAACATGTGTGATTCGCATGT...ATATGGCGGCCATATACGAGACT tinB-180

> res = motifEnrichment(sequences, PWMLogn.dm3.MotifDb.Dmel)
```

We read FASTA sequences using *Biostrings* function `readDNAStringSet`. Alternatively, sequence can be specified as a list of `DNAString` objects. See the package *Biostrings* for more information on handling FASTA sequences in R.

The function `motifEnrichment` is the top-level function for motif enrichment in single sequences and groups of sequences. It takes at least two input arguments: a set of sequences, and a set of PWMs. In this example, we used a lognormal background correction object `PWMLogn.dm3.MotifDb.Dmel` in place of PWMs so that lognormal correction is performed. The background correction object contains the background lognormal distribution parameters for 740 *D. melanogaster* MotifDb PWMs fitted on a set of 10031 *D. melanogaster* 2kb promoters.

The output of `motifEnrichment` is an object of class `MotifEnrichmentResults` that contains the scores for both sequences together (the ”group”) and individual sequences.

```
> res

An object of class 'MotifEnrichmentResults':
* created with 'affinity' scoring function with 'logn' background correction
* on a set of 2 sequence(s) and 740 PWMs
Result sets for the group: $group.nobg, $group.bg, $group.norm
Result sets for individual sequences: $sequence.nobg, $sequence.bg, $sequence.norm
Methods to extract data: motifRankingForGroup(), motifRankingForSequence()
Methods to plot data: plotTopMotifsGroup(), plotTopMotifsSequence()

> # PWMs enriched in both sequences (Lognormal P-value)
> head(motifRankingForGroup(res))

          da         dimm          twi          twi          tin           da 
0.0005281465 0.0005281465 0.0005583493 0.0005583493 0.0006296786 0.0015037152 

> # PWMs enriched in the first sequence (Lognormal P-values)
> head(motifRankingForSequence(res, 1))

         tin          vnd          vnd          tin          tin          vnd 
0.0003871013 0.0012951048 0.0012951048 0.0014255520 0.0014255520 0.0024719582 

> # PWMs enriched in the second sequence (Lognormal P-values)
> head(motifRankingForSequence(res, 2))

          da         dimm          twi          twi           da          tap 
0.0001420398 0.0001420398 0.0001974090 0.0001974090 0.0004429545 0.0004429545 
```

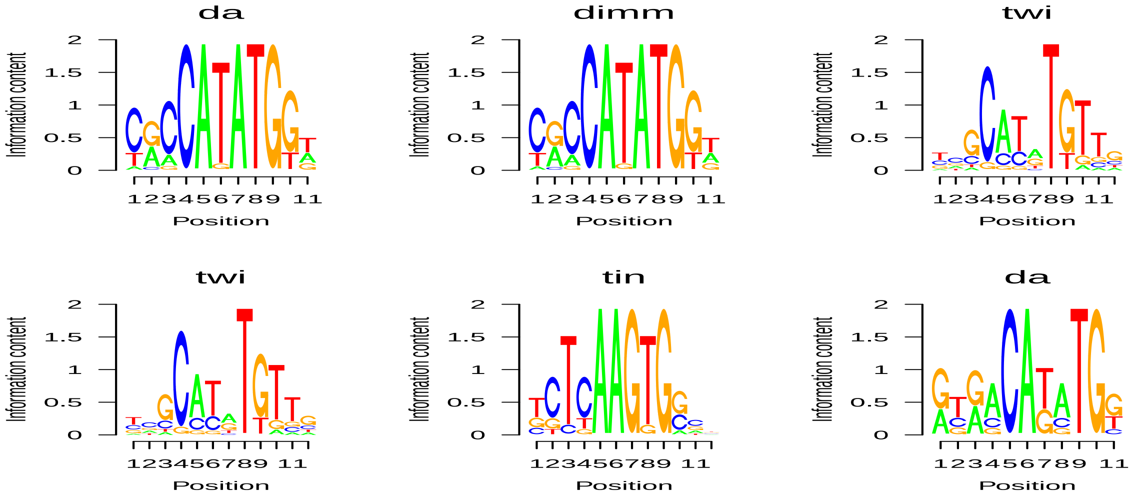
Some transcription factors have PWMs from multiple sources (which are sometimes identical, sometimes not) and from different multimers. We can refer back to the `MotifDb` package to find out more about the enriched motifs. We do this by looking at the original IDs of top ranked motifs:

```
> head(motifRankingForGroup(res, id=TRUE))

dimm_da_SANGER_5_FBgn0000413 dimm_da_SANGER_5_FBgn0023091 
                0.0005281465                 0.0005281465 
      twi_FlyReg_FBgn0003900                          twi 
                0.0005583493                 0.0005583493 
      tin_FlyReg_FBgn0004110  tap_da_SANGER_5_FBgn0000413 
                0.0006296786                 0.0015037152 

> library(MotifDb)
> db = values(MotifDb)
> db[db$providerName == "dimm_da_SANGER_5_FBgn0000413", "dataSource"]

[1] "FlyFactorSurvey"

> db[db$providerName == "dimm_da_SANGER_5_FBgn0023091", "dataSource"]
```

```
[1] "FlyFactorSurvey"
```

The first two motifs comes from FlyFactorSurvey and correspond to a heterodimer of Da and Dimm. We can further visualise the most enriched motifs through the package *seqLogo*. We plot the same 6 motifs:

```
> plotTopMotifsGroup(res, 6, rows=2, cols=3, xmargin.scale=0.7)
```



In the top 6 we recover the know motifs: twi and tin together with da and dimm that have similar motifs to twi.

Since background correction is performed after the raw individual scores are calculated, the resulting object always contains the raw values as well. We will illustrate this using a fixed-threshold background. Using a P-value threshold (e.g. 10^{-4}) we recover both the number of motif hits (with P-value $< 10^{-4}$) and the associated Z-score.

```
> data(PWMPvalueCutoff1e4.dm3.MotifDb.Dmel)
> res.count = motifEnrichment(sequences, PWMPvalueCutoff1e4.dm3.MotifDb.Dmel)
> #  PWMs sorted by number of motifs hits (P-value < 0.0001) in both sequences
> head(motifRankingForGroup(res.count, bg=FALSE))

    hth     twi     vis     twi   Bteb2 CG16778
      3       3       3       3       2       2

> #  PWMs sorted by the Z-score of motif enrichment
> head(motifRankingForGroup(res.count))

     twi      twi      Med       da     dimm       da
9.064881 9.064881 5.936686 5.923856 5.923856 5.923856

> # First sequence, PWMs sorted by number of motif hits (P-value < 0.0001)
> head(motifRankingForSequence(res.count, 1, bg=FALSE))

hth tin tin tin vis vnd
  2   2   2   2   2   2

> # Second sequence
> head(motifRankingForSequence(res.count, 2, bg=FALSE))

    twi     twi CG16778      da     dei      da
      3       3       2       2       2       2
```

Using the fixed-threshold approach also recovers the tin and twi motifs.

3 Parallel execution

Motif scanning is the most time consuming operation in all algorithms. Because of this, the package has a support for parallel motif scanning using the *parallel* core package. Note that parallel execution is currently not supported on Windows. To turn on parallel scanning, simply register a number of cores available to the package:

```
> registerCoresPWMEnrich(4)
```

After this command is executed, all further calls to *PWMEnrich* functions are going to be run in parallel using 4 cores (if possible). To turn off parallel execution call the function with parameter NULL:

```
> registerCoresPWMEnrich(NULL)
```

4 Using a custom set of PWMs

The package provides functions to read motifs in standard JASPAR and TRANSFAC formats. We will compile a lognormal background for a set of two de-novo motifs.

```
> motifs.denovo = readMotifs(system.file(package="PWMEnrich",
+   dir="extdata", file="example.transfac"), remove.acc=TRUE)
> motifs.denovo

$tin_like_motif
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
A   12    5    2    1    0   36   37    0    0     0     5     4     8    10
C   10    7   24    0   36    0    0    1    0     0     6    19     8     4
G   10   13    6    0    0    1    0   36    0    36    22     7     6     8
T    5   12    5   36    1    0    0    0   37     1     4     7    15    15

$gata_like_motif
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
A   17   17   13   42    0   42    0   42    0    21    12
C    7   12   19    0    0    0    0    0   42     5    16
G    6    6    7    0   42    0    0    0    0     8     5
T   12    7    3    0    0    0   42    0    0     8     9

> bg.denovo = makeBackground(motifs.denovo, organism="dm3", type="logn", quick=TRUE)
> res.denovo = motifEnrichment(sequences, bg.denovo)
> head(motifRankingForGroup(res.denovo))

 tin_like_motif gata_like_motif
    0.000199266     0.869277025
```

The function `makeBackground` does a couple of things. First, it converts the motifs into PWMs by taking the nucleotide background frequencies from the 10031 *D. melanogaster* promoters. Next it fits the lognormal parameters using the same set of sequence. In this example we used `quick=TRUE` for illustrative purposes. This fits the parameters quickly on a reduced set of 100 promoters. We strongly discourage the users to use this parameter in their research, and instead only use it to obtain rough estimates and for testing. The resulting object `bg.denovo` can be used same as before to perform motif enrichment (as demonstrated in the last two lines of the example).

The background object `bg.denovo` contains the two PWMs and their background distribution parameters. All of these can be accessed with the $ operator.

```
> bg.denovo
```

```
An object of class 'PWMLognBackground'
Background source: D.melanogaster (dm3) 100 unique 2kb promoters
Fitted on a mean sequence length of 988 for a set of 2 PWMs
Lognormal parameters: $bg.mean, $bg.sd
PWMS: $pwms
```

```
> bg.denovo$bg.mean
```

```
 tin_like_motif gata_like_motif 
      0.7653655       0.6743453 
```

5 Using a custom set of background sequences

Low-level functions are available for constructing custom backgrounds. We start with the two denovo motifs from previous section and fit the background to first 20 *D. melanogaster* promoters.

```
> library("BSgenome.Dmelanogaster.UCSC.dm3")
> # make a lognormal background for the two motifs using only first 20 promoters
> bg.seq = Dmelanogaster$upstream2000[1:20]
> # the sequences are split into 100bp chunks and fitted
> bg.custom = makePWMLognBackground(bg.seq, motifs.denovo, bg.len=100,
+     bg.source="20 promoters split into 100bp chunks")
> bg.custom
```

```
An object of class 'PWMLognBackground'
Background source: 20 promoters split into 100bp chunks
Fitted on a mean sequence length of 88 for a set of 2 PWMs
Lognormal parameters: $bg.mean, $bg.sd
PWMS: $pwms
```

The `makePWMLognBackground` function will convert frequency matrices (`motifs.denovo`) into PWMs using the same set of sequences on which the distribution is fitted. The frequency matrices can also be manually converted into PWMs using function `makePriors`, `getBackgroundFrequencies` and `PFMtoPWM`.

The resulting `bg.custom` object can be used as before for motif enrichment with `motifEnrichment` function (as described before).

6 Session information

- R version 3.0.0 (2013-04-03), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=C`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, utils
- Other packages: BSgenome 1.28.0, BSgenome.Dmelanogaster.UCSC.dm3 1.3.19, BiocGenerics 0.6.0, Biostrings 2.28.0, GenomicRanges 1.12.0, IRanges 1.18.0, MotifDb 1.2.0, PWMEnrich 2.2.0, PWMEnrich.Dmelanogaster.background 2.0.0
- Loaded via a namespace (and not attached): RCurl 1.95-4.1, Rsamtools 1.12.0, XML 3.96-1.1, bitops 1.0-5, evd 2.3-0, gdata 2.12.0.1, gtools 2.7.1, rtracklayer 1.20.0, seqLogo 1.26.0, stats4 3.0.0, tools 3.0.0, zlibbioc 1.6.0

References

Bodén, M. and Bailey, T. (2008). Associating transcription factor-binding site motifs with target go terms and target genes. *Nucleic acids research*, 36(12):4108–4117.

Frith, M. C., Fu, Y., Yu, L., Chen, J., Hansen, U., and Weng, Z. (2004). Detection of functional DNA motifs via statistical over-representation. *Nucl. Acids Res.*, 32(4):1372–1381.

Manke, T., Roider, H., and Vingron, M. (2008). Statistical modeling of transcription factor binding affinities predicts regulatory interactions. *PLoS computational biology*, 4(3):e1000039.

Stojnic, R. and Adryan, B. (2012). Identification of functional dna motifs using a binding affinity lognormal background distribution. *submitted*.

Stormo, G. (2000). Dna binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23.

Sui, S., Mortimer, J., Arenillas, D., Brumm, J., Walsh, C., Kennedy, B., and Wasserman, W. (2005). opossum: identification of over-represented transcription factor binding sites in co-expressed genes. *Nucleic acids research*, 33(10):3154–3164.