Using profileScoreDist

Pål O. Westermark
Carité University Medicine Berlin

Introduction

Position count matrices describe empirical frequencies encountered in short sequence motifs in biological sequences, usually DNA sequences. They are often used to describe transcription factor binding sites, and can be employed to predict binding sites in collections of sequences of interest. For this, typically log-odds scores for matches to a count matrix are computed for each possible location on the sequences of interest. This can be done with the `matchPWM` from the `Biostrings` package, for example. How to decide whether a score is large? Often, cutoffs such as 80% of the maximal value are employed. However, it may be desirable with more precise cutoffs. For example, a cutoff corresponding to a 1% probability for one or more false hits for a given sequence (false discovery rate). Or a cutoff corresponding to a 90% probability of discovering a sequence generated by the probabilities inherent in the count matrix, present in a sequence. Rahmann, Müller, and Vingron (2003) addressed this problem. This package bring this methodology to R/Bioconductor.

This vignette describes the capabilities of the profileScoreDist package. The package deals with 3 main tasks, each described in the article by Rahmann et al.

1. The function `regularizeMatrix()` adds pseudocounts to a position count matrix. It does so using a particularly careful method. Pseudocounts are drawn to match the overall nucleotide distribution of the count matrix, but are weighted for each position. The weights are smaller the more the count distribution for a given position deviates from the overall nucleotide distribution. In this way, fewer pseudocounts are added for positions with distinct signals. See the article by Rahmann et al. for more information.
2. The function `computeScoreDist()` computes probability distributions for the standard log-odds matrix scores, given nucleotide sequences assumed either to be
   - generated by a background distribution operating with a given GC probability independently for each position. From this, false discovery rates (FDRs) can be estimated (see task 3).
   - generated by the probabilities defined each position in the count matrix. From this distribution, False negative rates (FNRs) can be estimated (see task 3). The function also returns the scores associated with the distributions.
3. The function `scoreDistCutoffs()` estimates score cutoffs for specified FDRs, FNRs, or a tradeoff between these two, from the distributions and scores calculated in step 2. The tradeoff is specified as the point where $c \times \text{FDR} = \text{FNR}$, the user specifies the parameter c.

Step 2 is the computational bottleneck, due to the discrete convolution algorithm involved. The algorithm was therefore implemented in C++, and should be relatively optimized: the performance penalty imposed by R will be insignificant.

The user is assumed to have read the article by Rahmann, Müller, and Vingron (2003), in order to use the package and interpret the results properly.

A typical workflow would be to first regularize a position count matrix with `regularizeMatrix()`. Then, the distributions for the regularized matrix for an ensemble of GC percentages would be computed with `computeScoreDist()`. Finally, Score cutoffs for a given FDR (or FNR, or a tradeoff between the two) would be calculated with `scoreDistCutoffs()`.

Example

The following example code applies these 3 steps to reproduce the results reported by Rahmann et al. for the INR (or cap) count matrix.

We load the package

```
library(profileScoreDist)
```

The cap signal (or INR element) was analyzed by Rahmann et al. For convenience, this public domain count matrix is included in the package.

```
data(INR)
INR
##   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## A   49    0  288   26   77   67   45   50
## C   48  303    0   81   95  118   85   96
## G   69    0    0  116    0   46   73   56
## T  137    0   15   80  131   72  100  101
```

This matrix is regularized:

```
inr.reg <- regularizeMatrix(INR)
inr.reg
##        [,1]         [,2]         [,3]      [,4]         [,5]      [,6]
## A  49.60074   0.02837893 288.03350163  26.45590  77.16267735  89.37062
## C  48.82428 303.03893853   0.04596735  81.62553  95.22320845 148.69457
## G  69.35925   0.01697079   0.02003420 116.27263   0.09728214  59.37778
## T 137.63467   0.02998173  15.03539375  80.48164 131.17186511  95.63407
##        [,7]      [,8]
## A  46.55883  53.27450
## C  87.13886 100.49292
## G  73.93219  57.95817
## T 101.64687 104.45944
```

The weak C signal at position 6 gets more pseudocounts than the strong signal at position 2, as expected.

Probability distributions will be computed for discrete values of the possible log-odds scores. The granularity (intervals between discrete scores for which probabilities are computed) must be specified. Further, probability distributions should in practice be computed for different GC contents corresponding to the different sequences of interest. Here, we will compute distributions for 1%, 2%, up to 99% GC content.

```
granularity <- 0.05
gcgran <- 0.01
gcmin <- 0.01
gcmax <- 0.99
## gc fractions to consider
gcs <- seq(gcgran*round(gcmin/gcgran), gcgran*round(gcmax/gcgran), gcgran)
```

The distributions are then computed for each GC content:

```
## compute probability distributions
distlist <- lapply(gcs, function(x) computeScoreDist(inr.reg, x, granularity))
```

The profile score distribution is returned as a `profileScore` object. This object contains the signal and background distributions, as well as the scores. These are accessible with the `signalDist`, `backgroundDist`, and `score` methods, respectively.

A plot of the results for GC content 50% neatly reproduces Figure 1 in the article by Rahmann et al.

```
distlist[[50]]
##
## Profile score distribution
##
## Maximum score: 5.25
```

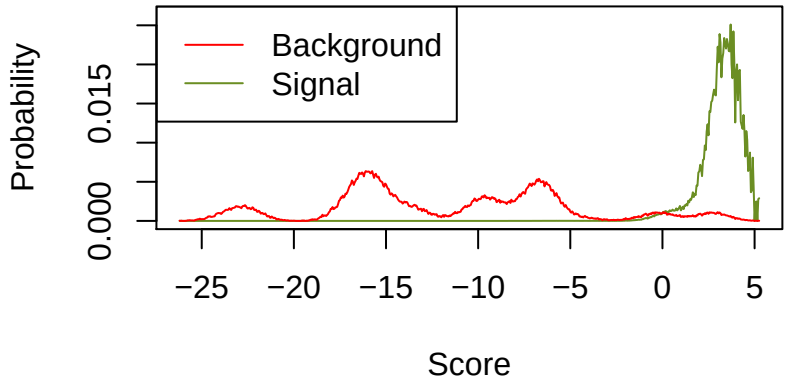

Figure 1: Reproduction of Figure 1 in the article by Rahmann et al.

```
## Minimum score: -26.2
## Number of discrete probabilities: 630
plotDist(distlist[[50]])
```

With the distributions for the possible scores, we may compute score cutoffs resulting in 5% FDR, or a 5% FNR. We also get the point where FDR=FNR (c=1):

```
ab5 <- scoreDistCutoffs(distlist[[50]], 500, 1, c=1, 0.05)

## 5% FDR
ab5$cutoffa
## [1] 4.95
## 5% FNR
ab5$cutoffb
## [1] 1.25
## FDR = FNR
ab5$cutoffopt
## [1] 4
```

These cutoffs may then be employed in downstream analysis.

References

Rahmann, Sven, Tobias Müller, and Martin Vingron. 2003. "On the Power of Profiles for Transcription Factor Binding Site Detection." *Statistical Applications in Genetics and Molecular Biology* 2: Article7. doi:10.2202/1544-6115.1032.