BCRANK: predicting binding site consensus from ranked DNA sequences

Adam Ameur
The Linnaeus Centre for Bioinformatics, Uppsala University, Sweden

November 1, 2008

1 Introduction

This document describes the `BCRANK` `R` package. `BCRANK`[1] is a method that takes a ranked list of genomic regions as input and outputs short DNA sequences that are overrepresented in some part of the list. The algorithm was developed for detecting transcription factor (TF) binding sites in a large number of enriched regions from high-throughput ChIP-chip or ChIP-seq experiments, but it can be applied to any ranked list of DNA sequences.

`BCRANK` uses a heuristic search strategy. First a score is computed for an initial short consensus sequence, typically selected at random. The score takes into account both the number of consensus occurrences and the rank of the genomic regions. Then all consensus sequences in a neighborhood of the start guess are evaluated and the one with highest score is kept as the starting point for the next iteration. When a local optimum is found, the algorithm is terminated and the locally optimal consensus is reported as a result. In order to increase the chance of detecting the globally optimal solution, the algorithm may be restarted several times using different random starting points. Alternatively, `BCRANK` can be used for assigning scores to previously established consensus sequences. The sections below describe in more detail how the neighborhood, scoring function and start guess are implemented.

`BCRANK` can extend and shorten motifs and can therefore be used in situations where the motif length is not known a priori. Moreover, `BCRANK` implements two optional penalties that can give relatively higher scores some consensus sequences. The effect of the penalties is that i) the consensus contains fewer redundant bases (i.e. other bases than A, C, G, T), and ii) the consensus will not be frequently occurring as a repetitive element in the enriched regions.

1.1 Neighborhood

In BCRANK, all consensus sequences are represented by IUPAC nucleotide symbols. The neighborhood of one consensus sequence s consists of all consensuses that can be generated from s by first adding one IUPAC letter N (representing any nucleotide) to either side of s and then flipping any base to any other IUPAC symbol. Since there are 15 symbols in total, a sequence of length l will have $14 \cdot (l + 2)$ neighbors. After each search step any flanking Ns are removed from the highest scoring sequence in the neighborhood. The removal and additions of flanking Ns allows the algorithm to shorten and extend the predicted binding sites.

1.2 Scoring function

The score tells whether a given consensus sequence is overrepresented in some part of the ranked list or not. Starting from N ranked regions and a consensus sequence c, a binary vector of size N is created, with 1 at position i if c is occurring in sequence number i, and 0 if not. The reverse complement of c is also allowed to match. Then the cumulative sum of the match vector is computed and stored in a vector called A. The A-vector tells where in the ranked list most occurrences are located (see Figure 1).

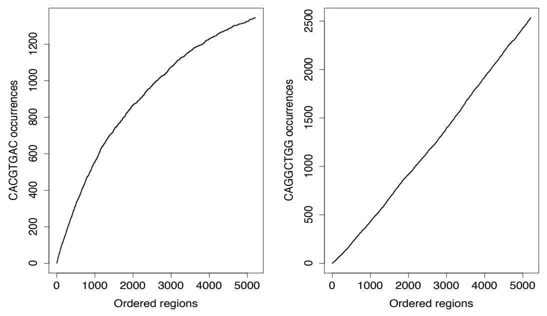


Figure 1: A-vectors for the two consensus sequences `CACGTGAC` (left) and `CAGGCTGG` (right). On the x-axis are the top 5211 regions from a whole genome ChIP-chip study on USF1 in human liver cells[2], ranked by their enrichment signal. The aim of `BCRANK` is to detect sequences that are biased towards some part of the list. Therefore `CACGTGAC` will get a higher score than `CAGGCTGG` even though it has a lower number of total occurrences. It is important to have enough number of ranked input regions to `BCRANK`, so this bias is observed for the correct binding motif. The established USF1 binding sequence is `CACGTG`.

To compute a score, A is compared to what it would look like if the genomic regions were randomly ordered. Therefore a large number R (the `reorderings` parameter to `bcrank()`) of random orderings of the input regions are generated, and a corresponding vector A_j is computed for each re-ordering $1 \leq j \leq R$ as above. For each j, the difference D_j between A_j and A is estimated by the area between the corresponding lines (see Figure 2). When calculating D_j, the A and A_j vectors are first scaled so they range between 0 and 1.

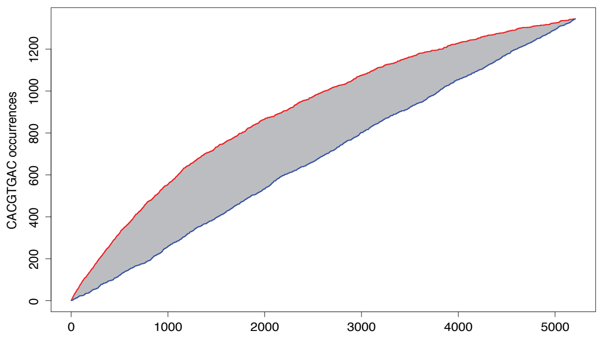


Figure 2: The vector A (red line) and a corresponding vector A_j (blue line) for `CACGTGAC` in the USF1 data. There is a clear bias towards the high scoring sequences as indicated by the red line. The significance of this bias can be estimated by comparing to the `CACGTGAC`-occurrences in randomly ordered regions, as indicated by the blue line. D_j corresponds to the grey area between the two lines.

D_j will be close to zero when the consensus occurrences are distributed as expected by random sampling. If on the other hand all D_j are far off from zero, c is biased towards some part of the list. Therefore the score is calculated as the t-statistic T for the D_j being drawn from a distribution centered around zero. Consensus sequences that are biased towards some part of the list will thus get high scores whereas consensuses with no bias will get low scores. Moreover, consensuses that are matching just a few regions will not get a high T even if it is matching only among the top ranked regions. This is because there will be a high variation within the D_j values which will result in a low T.

1.2.1 Penalties

The t-statistic gives consensus sequences that are biased towards some part of the list. But there may be other issues to take into account if the aim is to detect TFBS from ChIP-chip or ChIP-seq data. Therefore, `BCRANK` implements two optional penalties, $P1$ and $P2$, with values between 0 and 1. The final scoring function is defined as: $score = T \cdot P1 \cdot P2$. If a penalty is not used it will be set to 1.

- $P1$ - Penalty on non-specific bases. Let l be the length of the consensus sequence and b the total number of fixed bases (A, C, G, T) in the sequence. If there are no fixed bases, b is set to 0.5. The penalty is then defined as $P1 = b/l$.

- $P2$ - Penalty on repetitive motifs. Let $r_n, n \in 1, 2$ be the number of input DNA regions that contain at least n occurrences of the consensus. Then $P2 = 1 - (r_2/r_1)$.

1.3 Start guess

In case the algorithm is used for ab inito search, the initial guess is a randomly generated consensus of a specified length (the `length` parameter to `bcrank()`), with 10 bases as default. Multiple restarts with different random start guesses are usually required to increase the chance of finding the globally optimal solution. The number of restarts is determined by the `restarts` parameter.

`BCRANK` can also use start guesses passed to `bcrank()` by the `startguesses` parameter. By setting the `do.search` parameter to `FALSE`, `BCRANK` assigns scores for the given start guesses without performing any search.

1.4 Additional information

Some other important details:

- The algorithm randomly re-orders the data when the score is calculated. This implies that the same consensus sequence will get different `BCRANK` scores in the same data when run with different re-orderings. The variability in scores can be decreased by increasing the `reorderings` parameter

- The algorithm performs a breadth-first search, meaning that the highest scoring neighbor in the neighborhood is selected in each search step.

- The algorithm keeps track of all consensus sequences that have already been tested so the same sequence is not visited twice when performing a search.

2 BCRANK - An example run

The user is required to load the package using the `library()` command:

```
> library(BCRANK)
```

2.1 Sequence data

`BCRANK` takes a fasta file containing ranked sequences as input. The command below loads an example file containing 2500 ranked regions from a whole genome ChIP-chip experiment for the protein USF1 in the human liver cell line HepG2[2].

```
> fastaFile <- system.file("Exfiles/USF1_small.fa", package = "BCRANK")
```

2.2 Running BCRANK

The `bcrank()` function call below runs the `BCRANK` algorithm on the example USF1 data set. The `set.seed()` call sets seed for the random number generator for reproducibility.

```
> set.seed(0)
> BCRANKout <- bcrank(fastaFile, restarts=25, use.P1=TRUE, use.P2=TRUE)
```

Since it takes some time to run the algorithm, results can instead be loaded from a previous run on a larger USF1 data set containing the top 5211 regions:

```
> data(BCRANKout)
```

2.3 BCRANK output

An object of type `BCRANKresult` is returned:

```
> BCRANKout

An object of class "BCRANKresult"

Top 25 DNA motifs predicted by BCRANK:

     Consensus     Score
1     GTCACGTG 316.34270
2     CACGTGAC 304.59499
3       CGCGGA 147.04100
4       GCGAST 135.22207
5    AHATAATAA 128.92440
6     GCGGNGCG 121.87198
7    TNCDGGGCG 119.76454
8    GCGGGGVNG 119.65945
9    CCGCGNTBY 118.77481
10   GDGCGGHGH 115.19679
11   TNCGCGNDG 112.95972
12   CGGGNGMGC 111.93052
13  GNNDCTCGCS 111.12770
14 CGCGNNBCTBC 107.91403
```

```
15   CGGWSCVGA  93.88154
16   TNNCCAVCG  90.63345
17    GCACANAT  84.98763
18  KTNAGABCCT  84.08165
19   TATANBDAC  80.98513
20  AACAKADTDA  79.75340
21 CCVVDDGGACG  78.86994
22  CTCRATTGHT  74.67093
23 CTATKANYWMA  68.48067
24 CBCNCGHANTR  67.87154
25  KCRTKAHTDC  52.30902


Use methods toptable(object) and fname(object) to access object slots.
```

2.3.1 The `BCRANKsearch` object

Use the `toptable()` function to access information about each motif found by `bcrank`. It returns an object of type `BCRANKsearch`:

```
> topMotif <- toptable(BCRANKout, 1)
> topMotif
```

```
An object of class "BCRANKsearch"

Search path, starting from VGKTHVBRTB:

    Consensus      Score
1  VGKTHVBRTB   6.516874
2  CGKTHVBRTB  47.627369
3  CGKTHVBRTG  73.217992
4  CGKTCVBRTG  88.236721
5  CGKTCVCRTG 105.752733
6   GKTCVCRTG 142.234278
7   GKTCACRTG 188.156513
8   GGTCACRTG 235.594317
9   GGTCACGTG 255.725078
10   GTCACGTG 316.342696

Position weight matrix for search result (GTCACGTG):

  1 2 3 4 5 6 7 8
A 0 0 0 1 0 0 0 0
C 0 0 1 0 1 0 0 0
G 1 0 0 0 0 1 0 1
T 0 1 0 0 0 0 1 0


Use methods searchPath(object) and pwm(object) to access object slots.
```

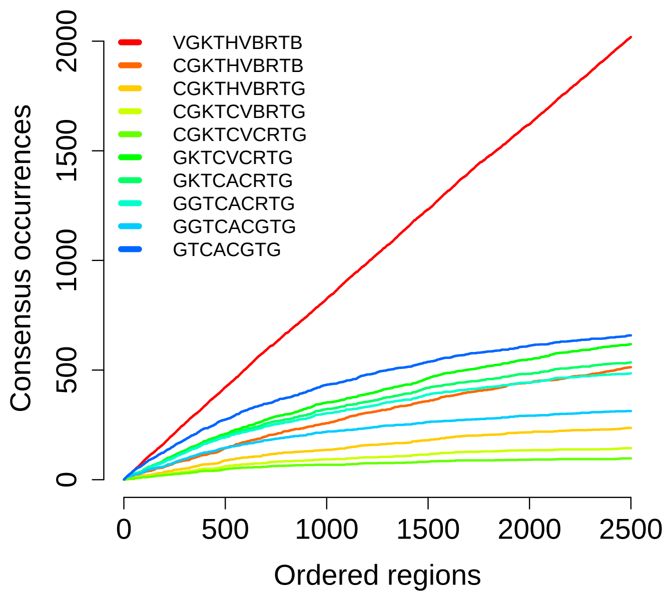
The pwm() function returns the position weight matrix for the search result. If normalize is set to TRUE each column will sum to 1. The weight matrix can be viewed as a sequence logo by using the seqLogo Bioconductor package.

```
> weightMatrix <- pwm(topMotif, normalize = FALSE)
> weightMatrix
```

```
    1   2   3   4   5   6   7   8
A   0   0   0 759   0   0   0   0
C   0   0 759   0 759   0   0   0
G 759   0   0   0   0 759   0 759
T   0 759   0   0   0   0 759   0
```

The search path can be visualized. For each consensus in the search path, the number of occurences among the ranked regions are plotted. As seen in the figure, BCRANK searches for consensus sequences that don't give straight lines.

```
> plot(topMotif)
```



2.3.2 Predicted binding sites

Individual predicted binding sites can be reported by the matchingSites function.

```
> topConsensus <- as.character(toptable(BCRANKout)[1, "Consensus"])
> print(topConsensus)
```

```
[1] "GTCACGTG"
```

```
> bindingSites <- matchingSites(fastaFile, topConsensus)
> nrSites <- nrow(bindingSites)
> cat("Number predicted binding sites:", nrSites, "\n")

Number predicted binding sites: 842

> print(bindingSites[1:15, ])

                 Region header Region nr Start  End Strand Sequence
1     8:71743620-71745120:4.234      seq1   513  520      + GTCACGTG
2     8:71743620-71745120:4.234      seq1   515  522      - CACGTGAC
3     9:94126661-94128161:3.921      seq2   795  802      + GTCACGTG
4    14:34660455-34661955:3.845      seq3   794  801      - CACGTGAC
5     3:20201537-20203037:3.752      seq4  1209 1216      + GTCACGTG
6     3:20201537-20203037:3.752      seq4  1273 1280      + GTCACGTG
7     3:20201537-20203037:3.752      seq4  1275 1282      - CACGTGAC
8     1:28430913-28432413:3.709      seq5  1202 1209      - CACGTGAC
9    17:45804496-45805996:3.659      seq6   834  841      + GTCACGTG
10   17:45804496-45805996:3.659      seq6   836  843      - CACGTGAC
11 6:151814107-151815607:3.636      seq7   927  934      - CACGTGAC
12   10:45541609-45543109:3.605      seq8  1036 1043      + GTCACGTG
13   10:45541609-45543109:3.605      seq8  1051 1058      + GTCACGTG
14   10:45541609-45543109:3.605      seq8  1038 1045      - CACGTGAC
15    6:33493266-33494766:3.552      seq9   833  840      + GTCACGTG
```

As seen in the example above, some binding sites can be reported both on the sense and antisense strands.If the consensus is palindromic, duplicate enries can be avoided by setting the `revComp` argument in the `matchingSites()` call to `FALSE`.

References

[1] Ameur, A., Rada-Iglesias, A., Komorowski, J., Wadelius, C. *Novel algorithm and ChIP-analysis identifies candidate functional SNPs.* Submitted

[2] Rada-Iglesias, A., Ameur, A., Kapranov, P., Enroth, S., Komorowski, J., Gingeras, T. R., Wadelius, C. *Whole-genome maps of USF1 and USF2 binding and histone H3 acetylation reveal new aspects of promoter structure and candidate genes for common human disorders.* Genome Res. 2008 Jan 29