BCRANK: predicting binding site consensus from ranked DNA sequences

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

September 2, 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. Once the algorithm finds a local optimum, 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.

The main aim of BCRANK is to predict the sequences that are bound by TF, and not necessarily to generate a perfect position weight matrix. Therefore BCRANK implements two penalties to give relatively higher scores to consensus sequences that are more likely to be bound by the TF. The effect of the penalties is that i) consensus sequences contain as few redundant bases as possible (i.e not in A, C, G, T), and ii) the consensus should not be frequently occurring as a repetitive element in the input 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 consensus sequences 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](image1.png)

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\cite{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 $A$ and $A_j$ (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](image2.png)

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 penalties, $P_1$ and $P_2$, with values between 0 and 1. The final scoring function is defined as: $score = T \cdot P_1 \cdot P_2$.

- $P_1$ - 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 $P_1 = b/l$.

- $P_2$ - 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 $P_2 = 1 - (r_2/r_1)$.

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 is non-deterministic because of the random re-orderings when the score is calculated. This also 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:

```r
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].

```r
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.

```r
set.seed(0)
BCRANKout <- bcrank(fastaFile, restarts=25)
```

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:

```r
data(BCRANKout)
```

## 2.3 BCRANK output

An object of type `BCRANKresult` is returned:

```r
BCRANKout
```

An object of class "BCRANKresult"

Top 25 DNA motifs predicted by BCRANK:

<table>
<thead>
<tr>
<th>Consensus</th>
<th>Scores</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TGGCAGG</td>
<td>316.34270</td>
<td>1</td>
</tr>
<tr>
<td>CTGTGTG</td>
<td>304.59499</td>
<td>2</td>
</tr>
<tr>
<td>CGCGGA</td>
<td>147.04100</td>
<td>3</td>
</tr>
<tr>
<td>GCGAST</td>
<td>135.22207</td>
<td>4</td>
</tr>
<tr>
<td>AHATAATAA</td>
<td>128.92440</td>
<td>5</td>
</tr>
<tr>
<td>GCGACG</td>
<td>121.87198</td>
<td>6</td>
</tr>
<tr>
<td>TNCGCGN</td>
<td>119.76454</td>
<td>7</td>
</tr>
<tr>
<td>GCGGGG</td>
<td>119.65945</td>
<td>8</td>
</tr>
<tr>
<td>CCCTG</td>
<td>118.77481</td>
<td>9</td>
</tr>
<tr>
<td>GCGGGH</td>
<td>115.19679</td>
<td>10</td>
</tr>
<tr>
<td>TNCGCGN</td>
<td>112.95972</td>
<td>11</td>
</tr>
<tr>
<td>CGGTG</td>
<td>111.93052</td>
<td>12</td>
</tr>
<tr>
<td>GNNDCTCG</td>
<td>111.12770</td>
<td>13</td>
</tr>
<tr>
<td>CCGNNBCTBC</td>
<td>107.91403</td>
<td>14</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Consensus</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 VGKTHVBRTB</td>
<td>6.516874</td>
</tr>
<tr>
<td>2 CGKTHVBRTB</td>
<td>47.627369</td>
</tr>
<tr>
<td>3 CGKTHVBRG</td>
<td>73.217992</td>
</tr>
<tr>
<td>4 CGKTCVBRG</td>
<td>88.236721</td>
</tr>
<tr>
<td>5 CGKTCVCRGT</td>
<td>105.752733</td>
</tr>
<tr>
<td>6 GKTVCVCRGT</td>
<td>142.234278</td>
</tr>
<tr>
<td>7 GKTCAVCRTG</td>
<td>188.156513</td>
</tr>
<tr>
<td>8 GTCACGRTG</td>
<td>235.594317</td>
</tr>
<tr>
<td>9 GTCACGRTG</td>
<td>255.725078</td>
</tr>
<tr>
<td>10 GTCACGRTG</td>
<td>316.342696</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>759</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>759</td>
<td>0</td>
<td>759</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>759</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>759</td>
<td>0</td>
<td>759</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>759</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>759</td>
<td>0</td>
</tr>
</tbody>
</table>

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

> plot(topMotif)

References
