Find genomic loci bound to transcription factors with CENTIPEDE

Kamil Slowikowski

2015-09-17

Abstract

This is a practical tutorial for running CENTIPEDE with DNase-Seq data. It explains how to prepare the data and how to run the analysis. The goal is to predict if a putative transcription factor binding site is actually bound or not. For details about the statistical models underlying the methods, please see (Pique-Regi, et al. 2011).

Contents

1 Introduction

Transcription factors (TFs) are proteins involved in the transcription of DNA to RNA. They bind to genomic DNA and regulate the transcription of nearby genes. Biologists can perform experiments to identify the specific sequences of nucleotides to which a TF can bind, called motifs. Computational biologists can use these motifs to identify sites in the genome where a a TF is likely to bind. We can use data from DNase-Seq

experiments that measure open chromatin across the entire genome to determine if a putative TF binding site actually has a bound TF.

CENTIPEDE is a computational method to infer if a region of the genome is bound by a particular TF. It uses information from a DNase-Seq experiment about the profile of reads surrounding a putative TF binding site. Further, it is able to incorporoate prior information such as sequence conservation across species. The method was created by Roger Pique-Regi and Jacob Degner when they were working in Jonathan Pritchard's group at University of Chicago in 2011.

2 Software requirements

1. MEME Suite
2. CENTIPEDE
3. Rsamtools
4. Bedtools

2.1 MEME Suite

Go to the MEME Suite download page to find the latest version of software:

http://meme-suite.org/doc/download.html

In this tutorial, I'll use version 4.10.1 patch 4:

```
wget http://meme-suite.org/meme-software/4.10.1/meme_4.10.1_4.tar.gz
cd meme_4.10.1
./configure --prefix=$HOME/meme --with-url="http://meme-suite.org"
make
make install
```

Add the `$HOME/meme/bin` folder to your `PATH` after you execute the above commands. You'll probably want to add this line to your `.bashrc` or similar.

```
export PATH="$PATH:$HOME/meme/bin"
```

2.2 CENTIPEDE

CENTIPEDE is an R package, so you must download and install R if you don't already have it installed.

Next, install CENTIPEDE with these commands in your shell (not in R):

```
wget http://download.r-forge.r-project.org/src/contrib/CENTIPEDE_1.2.tar.gz
R CMD INSTALL CENTIPEDE_1.2.tar.gz
```

Afterwards you should be able to run CENTIPEDE in an R session:

```
library(CENTIPEDE)
example(fitCentipede)
```

Finally, install the CENTIPEDE tutorial package:

```
install.packages("devtools")
devtools::install_github("slowkow/CENTIPEDE.tutorial")
```

2.3 Rsamtools

Rsamtools is an R package available via Bioconductor. Install it in an R session like this:

```
source("http://bioconductor.org/biocLite.R")
biocLite("Rsamtools")
```

2.4 Bedtools

Follow the instructions here to install Bedtools: https://bedtools.readthedocs.org/en/latest/content/installation.html

3 Input data

To complete this tutorial, you will need three inputs:

1. A position weight matrix (PWM) for a transcription factor (TF).
2. Genomic sequence for the organism of interest.
3. DNase-Seq data for the cell type of interest.

3.1 Position weight matrix

You can download thousands of motifs for many different organisms, collated from multiple different databases, all bundled in a single archive from the MEME Suite webpage if you click "Motif Databases" here: http://meme-suite.org/doc/download.html

In this tutorial, we'll just focus on a single PWM for the human STAT4 gene, taken from version 1.02 of the CISBP database:

```
wget http://cisbp.ccbr.utoronto.ca/data/1.02/DataFiles/PWMs/Files/M6496_1.02.txt

cat M6496_1.02.txt
```

```
Pos A   C   G   T
1   0.152046783625731   0.233918128654971   0.087719298245614   0.526315789473684
2   0.0 0.0 0.0409356725146199  0.95906432748538
3   0.0 0.0467836257309942  0.0 0.953216374269006
4   0.198830409356725   0.666666666666667   0.0467836257309942  0.087719298245614
5   0.181286549707602   0.725146198830409   0.0 0.0935672514619883
6   0.257309941520468   0.514619883040936   0.192982456140351   0.0350877192982456
7   0.701754385964912   0.192982456140351   0.105263157894737   0.0
8   0.0 0.0409356725146199  0.853801169590643   0.105263157894737
9   0.95906432748538    0.0 0.0409356725146199  0.0
10  1.0 0.0 0.0 0.0
11  0.514619883040936   0.105263157894737   0.233918128654971   0.146198830409357
12  0.578947368421053   0.175438596491228   0.0467836257309942  0.198830409356725
13  0.244152046783626   0.25    0.402046783625731   0.103801169590643
```

You can use the `matrix2meme` utility provided in the MEME Suite to create a file in MEME format.

```
matrix2meme < <(tail -n+2 M6496_1.02.txt | cut -f2-) > M6496_1.02.meme

cat M6496_1.02.meme
```

```
MEME version 4.4

ALPHABET= ACGT

strands: + -

Background letter frequencies (from uniform background):
A 0.25000 C 0.25000 G 0.25000 T 0.25000

MOTIF 1 TTTCCVAGAAAAN

letter-probability matrix: alength= 4 w= 13 nsites= 20 E= 0
  0.152047    0.233918    0.087719    0.526316
  0.000000    0.000000    0.040936    0.959064
  0.000000    0.046784    0.000000    0.953216
  0.198830    0.666667    0.046784    0.087719
  0.181287    0.725146    0.000000    0.093567
  0.257310    0.514620    0.192982    0.035088
  0.701754    0.192982    0.105263    0.000000
  0.000000    0.040936    0.853801    0.105263
  0.959064    0.000000    0.040936    0.000000
  1.000000    0.000000    0.000000    0.000000
  0.514620    0.105263    0.233918    0.146199
  0.578947    0.175439    0.046784    0.198830
  0.244152    0.250000    0.402047    0.103801
```

3.2 Genomic sequence

We'll use the UCSC human reference genome version hg19. You can download the reference genome here, or follow these commands:

```
wget "http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFaMasked.tar.gz"
tar -xzvf chromFaMasked.tar.gz
gunzip -c chr*.fa.masked > hg19.fa
```

3.3 DNase-Seq data

We'll use DNase-Seq data from human fibroblast cells downloaded from the ENCODE Project portal. In order to use this data with CENTIPEDE, we need two things:

1. The mapped reads in BAM format, so we can count the number of read starts at each genomic position.
2. Called peaks in ENCODE narrowPeak format, indicating genomic loci where there is enrichment of signal based on pooled normalized data.

Download the files:

```
# 2.12 MB
wget --no-check-certificate \
  https://www.encodeproject.org/files/ENCFF001UUQ/@@download/ENCFF001UUQ.bed.gz

# 4.4 GB
wget --no-check-certificate \
  https://www.encodeproject.org/files/ENCFF000SHS/@@download/ENCFF000SHS.bam
```

4 Analysis

4.1 Find putative TF binding sites

Select the most statistically significant DNase-Seq peaks with P < 1e-8:

```
dnase=ENCFF001UUQ.narrowPeak.gz
dnase_gt8=ENCFF001UUQ_gt8.narrowPeak.gz

zcat $dnase | awk '{if ($8 > 8) print}' | gzip > $dnase_gt8
```

Obtain nucleotide sequences within these peaks:

```
genome=hg19.fa
dnase_fasta=ENCFF001UUQ_gt8.fa

bedtools getfasta -fi $genome -bed $dnase_gt8 -fo $dnase_fasta
```

Search for sequences within these peaks that match the PWM:

```
meme=M6496_1.02.meme
sites=M6496_1.02.fimo.txt.gz

fimo --text --parse-genomic-coord $meme $dnase_fasta | gzip > $sites

zcat $sites | head
```

```
#pattern name   sequence name   start   stop    strand  score   p-value q-value matched sequence
1       chr1    753116  753128  +       13.53   1.14e-05                TTTCCCAGAAGGA
1       chr1    876297  876309  -       12.07   3.73e-05                CTTCCCCGAAGGG
1       chr1    1365583 1365595 -       11.88   4.24e-05                TTTCCAAGAAAGT
1       chr1    1365977 1365989 -       12.72   2.24e-05                CTTCCCAGGAGAG
1       chr1    1406805 1406817 -       11.2    6.73e-05                CTTCACAGAATTA
1       chr1    1566458 1566470 +       13.99   7.75e-06                TTTCCAAGAACCG
1       chr1    1837701 1837713 -       11.6    5.15e-05                TTTTTCAGAAAAC
1       chr1    1841434 1841446 -       10.75   9.12e-05                TTTCTGAGAAAGG
1       chr1    1841436 1841448 +       13.04   1.77e-05                TTTCTCAGAAACA
```

4.2 Determine if TF sites are bound

Start an R session and load the code provided in the package that accompanies this tutorial:

```
# install.packages("devtools")
library(devtools)
library(Rsamtools)
```

```
## Loading required package: S4Vectors
## Loading required package: stats4
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
##
## The following object is masked from 'package:stats':
##
##     xtabs
##
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, as.vector, cbind,
##     colnames, do.call, duplicated, eval, evalq, Filter, Find, get,
##     intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rep.int, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unlist, unsplit
##
## Loading required package: IRanges
## Loading required package: GenomeInfoDb
## Loading required package: GenomicRanges
## Loading required package: XVector
## Loading required package: Biostrings
```

```
library(CENTIPEDE)

# Install the tutorial package:
# devtools::install_github("slowkow/CENTIPEDE.tutorial")

# Load the functions and example data:
library(CENTIPEDE.tutorial)
```

Count read start positions within 100 bp upstream or 100 bp downstream of the 13 bp PWM match sites that were assigned P < 1e-4 by FIMO.

Note: The `cen` object from this step is included in the package, so you can skip this step to save time. If you wish to analyze other data or motifs, then you should call the `centipede_data` function on your own files.

```
cen <- centipede_data(
  bam_file = "ENCFF000SHS.bam",
  fimo_file = "M6496_1.02.fimo.txt.gz",
  log10p = 4,
  flank_size = 100
)
```

The `cen` object is a list with two elements:

1. `regions` is a dataframe with one row for each PWM region.
2. `mat` is a matrix with read counts for each PWM region.

Here are the selected PWM sites, including 100 bp flanks:

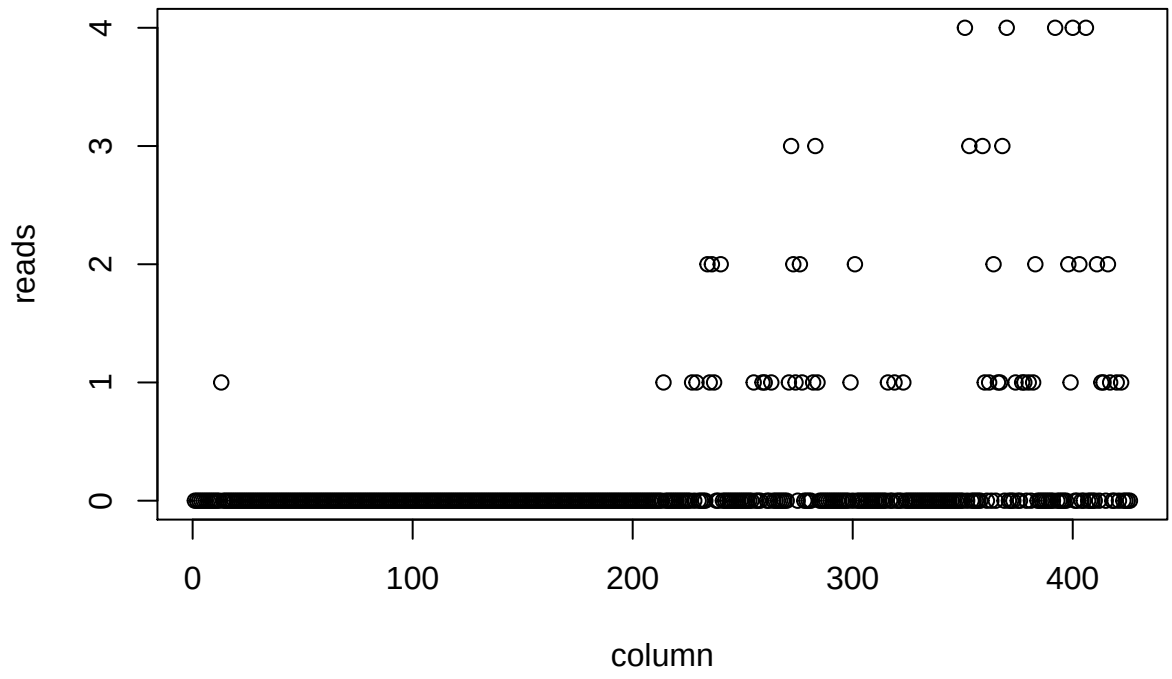
```
head(cen$regions)
```

```
##     sequence.name   start    stop X.pattern.name strand score  p.value
## 307          chr1  753016  753228              1      + 13.53 1.14e-05
## 315          chr1  876197  876409              1      - 12.07 3.73e-05
## 29           chr1 1365483 1365695              1      - 11.88 4.24e-05
## 30           chr1 1365877 1366089              1      - 12.72 2.24e-05
## 31           chr1 1406705 1406917              1      - 11.20 6.73e-05
## 64           chr1 1566358 1566570              1      + 13.99 7.75e-06
##     q.value matched.sequence
## 307      NA    TTTCCCAGAAGGA
## 315      NA    CTTCCCCGAAGGG
## 29       NA    TTTCCAAGAAAGT
## 30       NA    CTTCCCAGGAGAG
## 31       NA    CTTCACAGAATTA
## 64       NA    TTTCCAAGAACCG
```

In the count matrix `mat`, reads on the positive strand are counted in the first 213 columns of the matrix, and reads on the negative strand are counted in the last 213 columns of the matrix.

Below, we can see the read start counts for the first region `chr1:753016-753228`. Notice that there is just 1 read on the positive strand about 90 bases upstream of the PWM match site (columns 1-213 in the count matrix). On the negative strand, we have up to 4 reads mapping at single position (columns 214-426 in the count matrix).

```
plot(cen$mat[1,], xlab = "column", ylab = "reads")
```



We can see how many read starts occur in each region:

```
rowSums(cen$mat)[1:10]
```

```
##   chr1:753016-753228   chr1:876197-876409 chr1:1365483-1365695
##                   93                  120                   48
## chr1:1365877-1366089 chr1:1406705-1406917 chr1:1566358-1566570
##                  464                  161                  157
## chr1:1837601-1837813 chr1:1841334-1841546 chr1:1841336-1841548
##                  234                   38                   38
## chr1:2106319-2106531
##                   34
```

Finally, we can use CENTIPEDE to compute the posterior probability that a TF is bound at each peak:

```
library(CENTIPEDE)

fit <- fitCentipede(
  Xlist = list(DNase = cen$mat),
  Y = as.matrix(data.frame(
    Intercept = rep(1, nrow(cen$mat))
  ))
)
```

```
## Warning in cor(LogRatios, PriorLogRatio): the standard deviation is zero
```

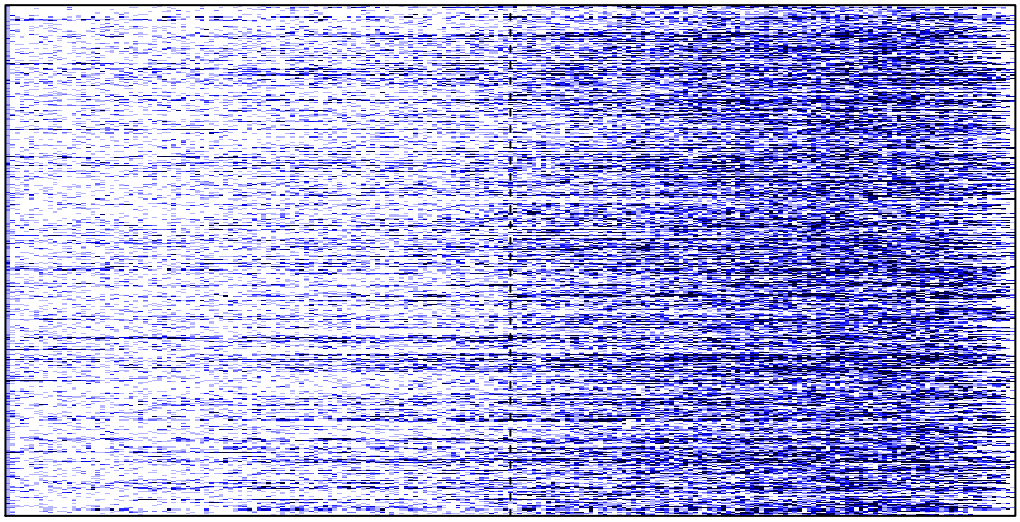
How many sites have a posterior probability of 1?

```
sum(fit$PostPr == 1)
```

```
## [1] 501
```

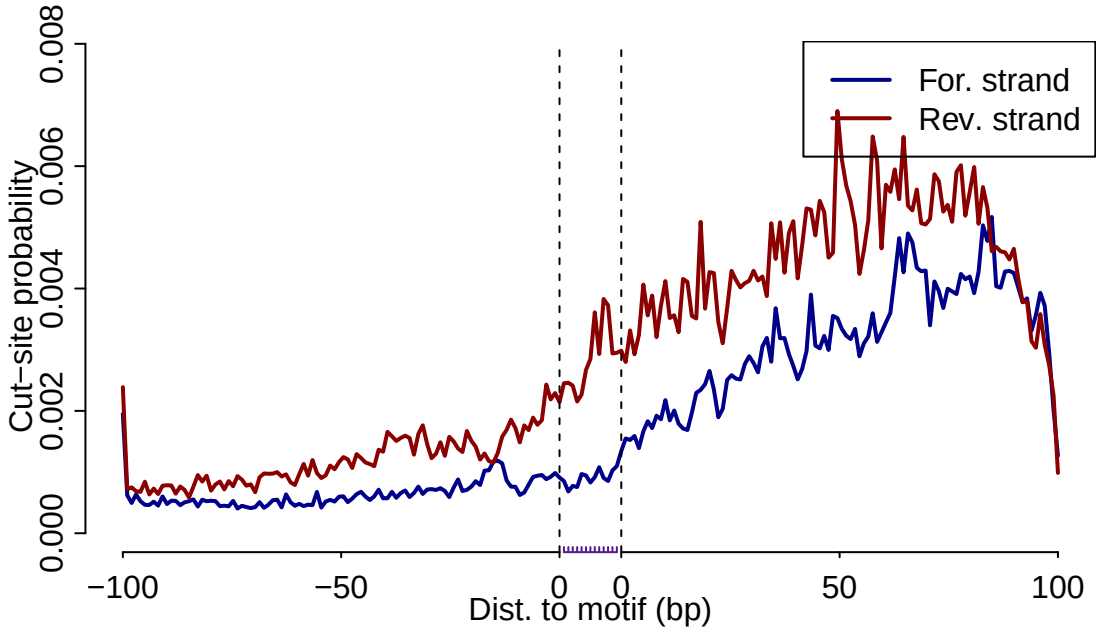
Plot a heatmap of the count matrix for sites predicted to be bound by STAT4:

```
imageCutSitesCombined(cen$mat[fit$PostPr == 1,])
```



Plot the footprint of STAT4 estimated by CENTIPEDE:

```
plotProfile(fit$LambdaParList[[1]], Mlen = 13)
```



4.3 Incorporate sequence conservation information

Let's run CENTIPEDE again, but this time we'll incorporate sequence conservation across multiple species. Will this information help to distinguish sites with or without a bound TF?

Download the conservation data from UCSC:

```
# 5.4 GB
wget \
http://hgdownload.cse.ucsc.edu/goldenpath/hg19/phastCons100way/hg19.100way.phastCons.bw
```

We need the `bigWigToBedGraph` utility to work with bigWig files:

```
wget http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/bigWigToBedGraph

# Put the executable into a folder that is listed in your PATH.
mv bigWigToBedGraph ~/bin
```

Extract the conservation information within DNase peaks:

```
cons=hg19.100way.phastCons.bw
cons_bed=${dnase_bed%%.*}_phastCons.bed.gz

bigWigRegions() {
```

```
  bw="$1"
  bed="$2"
  IFS=$'\t'
  while read chrom beg end rest; do
    # Write temporary files to RAM.
    out="/dev/shm/bigWigRegions_${USER}_${$}_${chrom}_${beg}_${end}"
    bigWigToBedGraph -chrom=$chrom -start=$beg -end=$end "$bw" "$out"
  done < "$bed"
  # Print the temporary files to stdout and then delete them.
  cat /dev/shm/bigWigRegions_${USER}_${$}_* | sort -k1V -k2n -k3n
  rm -f /dev/shm/bigWigRegions_${USER}_${$}_*
}

bigWigRegions $cons <(zcat $dnase_bed) | gzip > $cons_bed
```

Compute mean conservation scores for each PWM site:

Note: The `site_cons` object from this step is included in the package, so you can skip this step to save time if you're following along with the data from the tutorial.

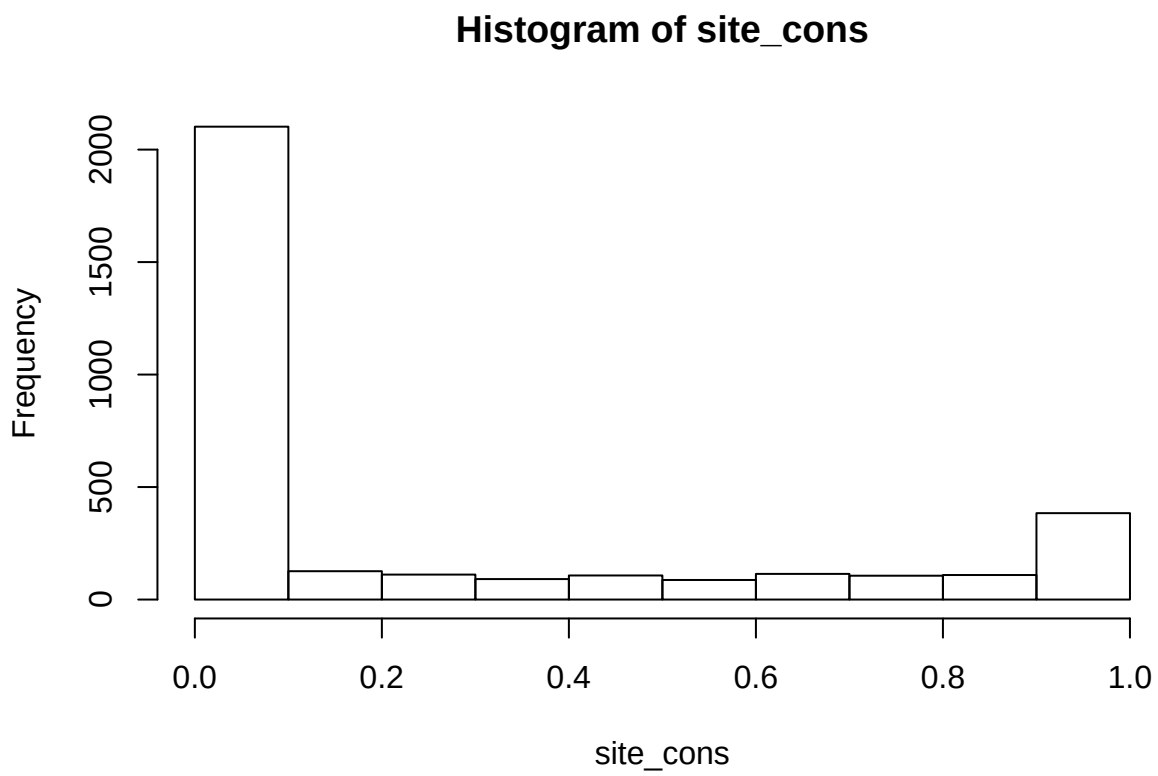
```
# Conservation scores for each base in the significant DNase peaks.
cons <- read_bedGraph('ENCFF001UUQ_gt8_phastCons.bed.gz')

# Get the 13 bp match sites without 100 bp flanks.
flank_size <- 100L
sites <- GRanges(
  seqnames = Rle(cen$regions$sequence.name),
  ranges = IRanges(
    start = cen$regions$start,
    end = cen$regions$stop
  ),
  strand = Rle(cen$regions$strand)
)
sites <- resize(sites, width(sites) - flank_size, fix = "end")
sites <- resize(sites, width(sites) - flank_size, fix = "start")

# Get the mean conservation score for each PWM binding site.
xs <- findOverlaps(sites, cons)
site_cons <- sapply(1:length(sites), function(i) {
  # Conservation scores for each positions in a PWM match.
  ys <- cons[subjectHits(xs[queryHits(xs) == i])]
  vals <- rep(ys$score, width(ys))
  idx <- seq(
    from = start(sites[i]) - min(start(ys)) + 1,
    length.out = width(sites[i])
  )
  vals <- vals[idx]
  mean(vals)
})
```

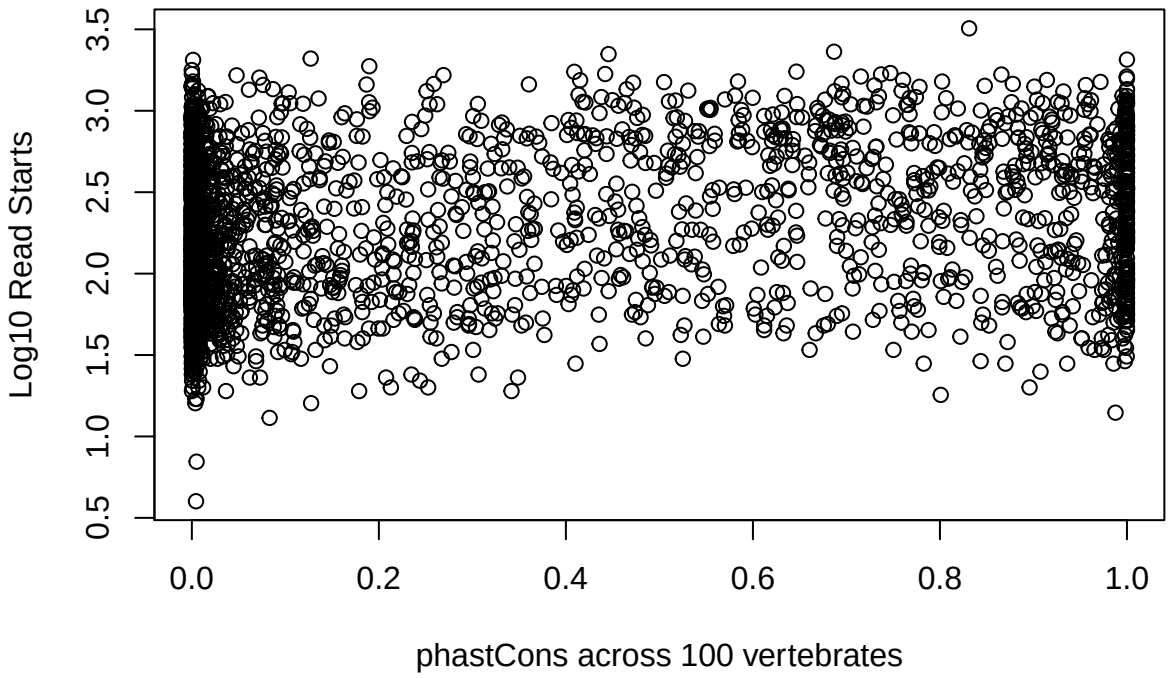
Some sites are much more highly conserved than others:

```
hist(site_cons)
```



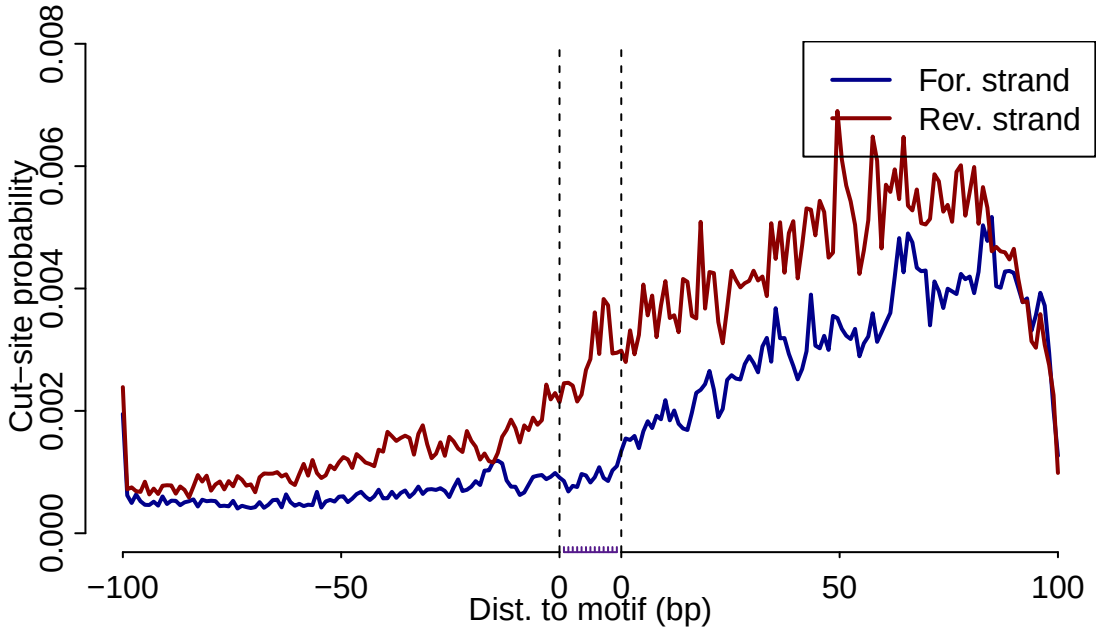
There is no apparent relationship between the number of read starts near PWM sites and the mean conservation score for the PWM site:

```
plot(site_cons, log10(rowSums(cen$mat) + 1),
     ylab = "Log10 Read Starts",
     xlab = "phastCons across 100 vertebrates")
```



Run CENTIPEDE again, but this time incorporate conservation information:

```
fit2 <- fitCentipede(
  Xlist = list(DNase = cen$mat),
  Y = as.matrix(data.frame(
    Intercept = rep(1, nrow(cen$mat)),
    Conservation = site_cons
  ))
)
plotProfile(fit2$LambdaParList[[1]], Mlen = 13)
```



Are the sites with probability 1 identical?

```
all.equal(fit2$PostPr == 1, fit$PostPr == 1)
```

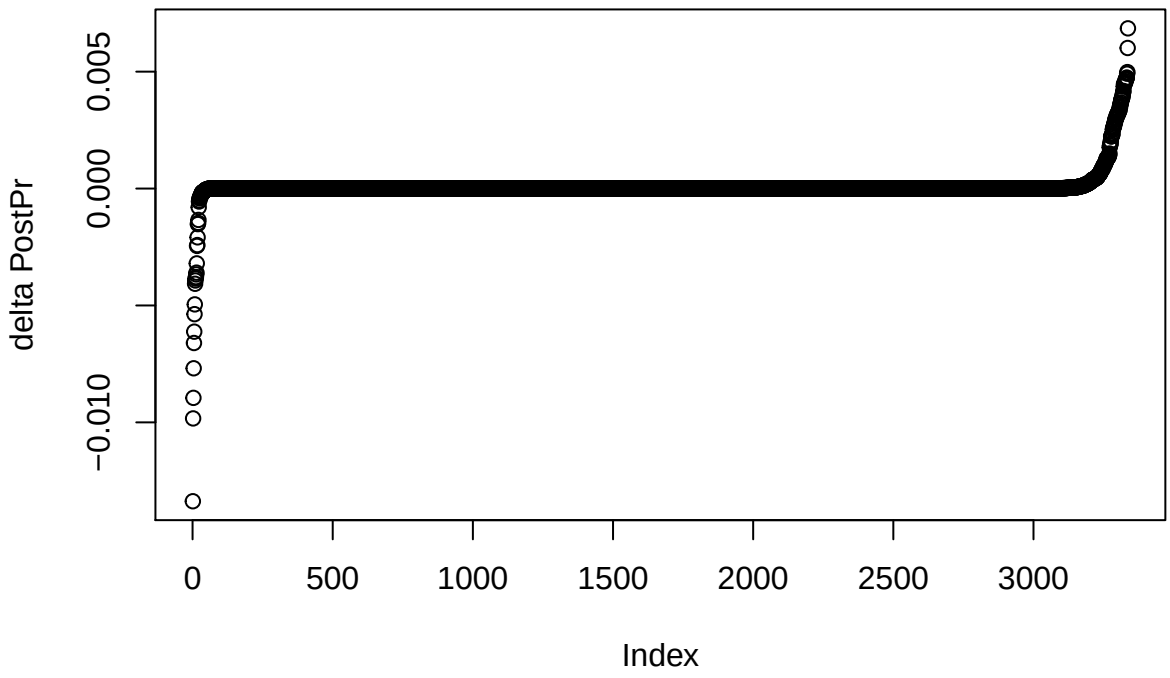
```
## [1] TRUE
```

After incorporating conservation scores, the posterior probabilities have remained nearly unchanged. The greatest increase in posterior probability attributable to incorporation of the conservation score is 0.00685.

```
range(fit2$PostPr - fit$PostPr)
```

```
## [1] -0.013373104  0.006851803
```

```
plot(sort(fit2$PostPr - fit$PostPr), ylab = "delta PostPr")
```



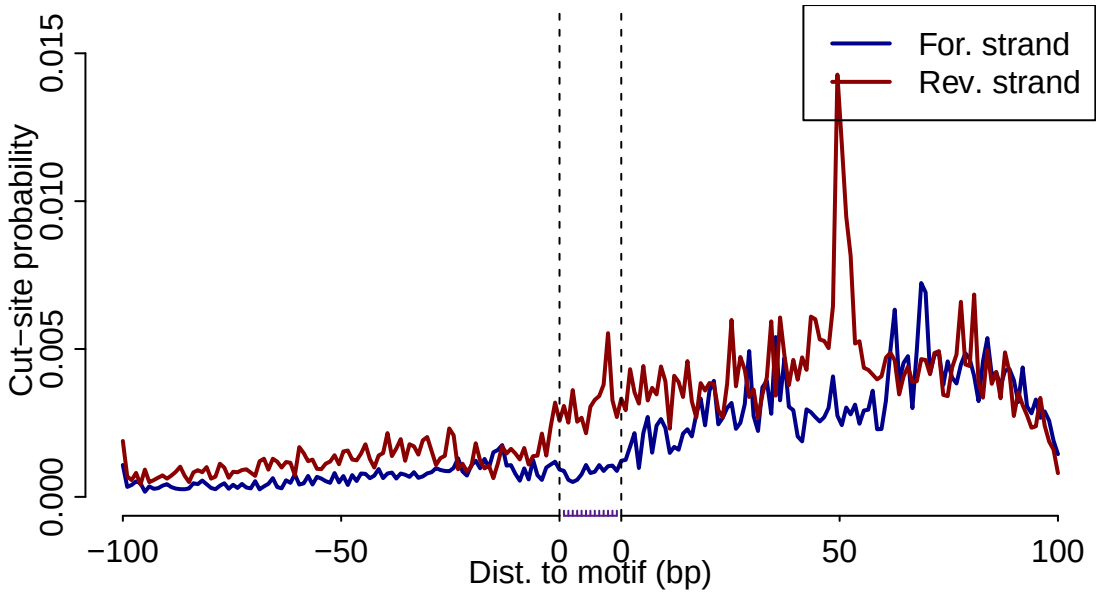
4.4 Restrict analysis to conserved sites

In addition to incorporating conservation information into the CENTIPEDE model, lets also use that information as a filter to limit the number of sites that are modeled by CENTIPEDE.

```
idx <- site_cons > 0.8
fit3 <- fitCentipede(
  Xlist = list(DNase = cen$mat[idx, ]),
  Y = as.matrix(data.frame(
    Intercept = rep(1, nrow(cen$mat[idx, ])),
    Conservation = site_cons[idx]
  ))
)
```

Now we can see that there might be an interesting signal 50 bp downstream of the PWM match sites. Perhaps this offset is in part due to the length of sequencing reads in this experiment (21 bp).

```
plotProfile(fit3$LambdaParList[[1]], Mlen = 13)
```



5 References

Pique-Regi, R. et al. Accurate inference of transcription factor binding from DNA sequence and chromatin accessibility data. Genome Res. 21, 447–455 (2011).