DEScan

John Koberstein, Bruce Gomes

Contents

1 Overview

This document describes how to use DEScan to detect regions of differential enrichment in epigenomic sequencing data. DEScan is an R/Bioconductor based tool, developed for Bioconductor v3.4.

2 A typical differential enrichment analysis workflow

2.1 Installation

DESCan is currently hosted on GitHub, so installation will make use of the devtools package.

```
library(devtools)
```

Use the install_github function to install DEScan. The repo location and a personal access token (PAT) need to be supplied.

```
install_github(repo = "jnkoberstein/DEScan",
               auth_token = "293d0010ce53d248175cee2d67ff248395f1a0a2")
```

```
library(DEScan)
library(RUVSeq)
library(EDASeq)
```

2.2 Example data

This analysis will make use of mouse Sono-seq data from mouse chromosome 19. This data was obtained from hippocampus of mice following learning through contextual fear conditioning. The experiment has 4 fear conditioned (FC) and 4 homecage control (HC) replicates. The data was aligned to the mouse genome (mm9) using bowtie2, allowing for multi-mapping reads. Duplicates were removed if present in isolation from duplicated reads relative to genomic location. Aligned data in this example is provided as bed files, but DEScan accepts bam format alignments.

```
bed.files <- list.files(system.file("extdata/Bed", package = "DEScan"),
                        full.names = T)
```

2.3 Calling peaks for each sample

First, peak calling for the samples will be done using the *findPeaks* function. Bed files are used as input in this case, but bam files work as well. Should bed files be used they need to be split up by chromosome in order to be loaded into R. We will perform this step on a single sample in the interest of time; however, note that a vector of file names can be supplied. The output of this function will be saved as an ".RData" object in a "Peaks" directory with subdirectories for each chromosome. findPeaks implements an adaptive window size scan to find peaks, and requires 2 parameters to define the size of the overlapping windows to be tested for enrichment. Enrichment for each window is calculated relative to 5kb, 10 kb (local) or the whole chromosome, and the maximum among the three is reported.

Parameter description:

chr Chromosome number, if multiple chromosomes are present in the alignment file the range should be defined, e.g. 1:19
filetype Format of the alignment file (bed or bam), default bam
fraglen Length of fragment sequenced, default 200.
rlen Length of read sequenced, no default. It has to be specified.
min_bin Size in bp of the minimum window size to scan. This parameter should be smaller if working with transcription factor binding ChIP-seq data
max_win Multiplier of the min_bin that defines the maximum window to scan, default 20.
blocksize How much of the chromosome will be analyzed at a time to avoid memory issues, default 10,000
ztresh Z-score threshold for reporting peaks, default 5 (we recommend keeping this low as it can be made more stringent later)
min_count A small integer added to the counts to prevent problems with log 0, default 0.1
output_name Name of the folder to save the Peaks (optional), if the directory doesn't exist it will be created. Default is "Peaks"

```
peaks <- findPeaks(bed.files[1], chr = 19, filetype = "bed", fraglen = 200,
                   rlen = 100, min_bin = 50, max_win = 20, blocksize = 10000,
                   zthresh = 5, min_count = 0.1, verbose = FALSE, save = FALSE)
head(peaks)
##      [,1]    [,2]      [,3]      [,4]
## [1,] "chr19" "3388939" "3389239" "75.1286531608231"
## [2,] "chr19" "3188289" "3188289" "42.2135146624493"
## [3,] "chr19" "3323339" "3323339" "40.1190053856675"
## [4,] "chr19" "3325389" "3325389" "39.6075238362831"
## [5,] "chr19" "3128389" "3128389" "31.9246068494623"
## [6,] "chr19" "3350639" "3350639" "30.5341293459834"
```

2.4 Aligning peaks across replicates to produce regions

After *findPeaks* has been run on each chromosome and each sample, the *finalRegions* function can be used to align overlapping peaks found in multiple samples. Peak files for all the alignment files can be found in the "extdata/Peaks" folder. *finalRegions* will produce one file containing the location of the aligned peaks for all chromosomes.

Parameter description:

zthresh Z-score for considering a peak for alignment, it needs to be equal or higher to the ztresh used in findPeaks, default 20.
min_carriers Minimum number of biological replicates required to overlap after aligning peaks for reporting, default 2
save_file Format of the output (bed or RData), default bed

```
peak.path <- system.file("extdata/Peaks", package = "DEScan")

regions <- finalRegions(peak_path = peak.path, chr = 19, zthresh = 20,
                        min_carriers = 4, save_file = "bed", verbose = FALSE)
head(regions)
##      Chr   Start     End  AvgZ NumCarriers
## 30 chr19 3282515 3283310 26.81           4
## 38 chr19 3323066 3324266 41.98           5
## 39 chr19 3325003 3325644 42.43           6
## 55 chr19 3387730 3389509 71.07           8
## 78 chr19 3485865 3487389 25.26           6
## 92 chr19 3530989 3531430 25.55           5
```

The output of this function is a bed-like file with columns indicating genomic coordinates as well as additional columns: AvgZ, average z-score of the peaks combined to form a common region, and NumCarriers, the number of samples a region was present in.

2.5 Counting reads in the final regions

The resulting regions can then be used to generate a count matrix using the *countFinalRegions* function. This function takes the regions to count across (can be any bed like data structure), and the path to bam files which contain the reads to be counted. Bam files for all the alignment files can be found in the "extdata/Bam/chr19" folder. The minimum number of carriers can also be specified in order to speed up the process. In this case we will not specify a minimum number of carriers and will filter after counting. This function is a wrapper for *featureCounts*.

```
region.file <- system.file("extdata/Regions", "FinalRegions_allChr.bed",
                           package = "DEScan")
bam.files <- system.file("extdata/Bam/chr19", package = "DEScan")

count <- countFinalRegions(region_file = region.file,
                           bam_file_path = bam.files,
                           min_carriers = 1, verbose = F)
```

The resulting count matrix contains a row for each region and a column for each sample. This structure is analogous to common RNA-seq data and can be normalized and analyzed with similar tools. First, we will rename and reorder the columns for readability and filter for a minimum of 4 carriers in order to only test relevant regions.

```
count <- count[regions$NumCarriers >= 4, ]
count <- count[rowSums(count) > 0,]
colnames(count) <- c("FC1", "FC4", "HC1", "HC4", "FC6", "FC9", "HC6", "HC9")
count <- count[,order(colnames(count))]
head(count)
##                        FC1 FC4 FC6 FC9 HC1 HC4 HC6 HC9
## chr19:3102794-3103194   5   3   4   5  10   4 130   4
## chr19:3107766-3108166  37   3 246  32  17  40   2   8
## chr19:3121794-3122194  15   1   5  53  10   3 110   0
## chr19:3128189-3128589 147   5  11   4  26  13   0   3
## chr19:3129060-3129460  18   6  29  30  10 149   0   6
## chr19:3151966-3152366  28   7 232  34  44  46   0   6
```

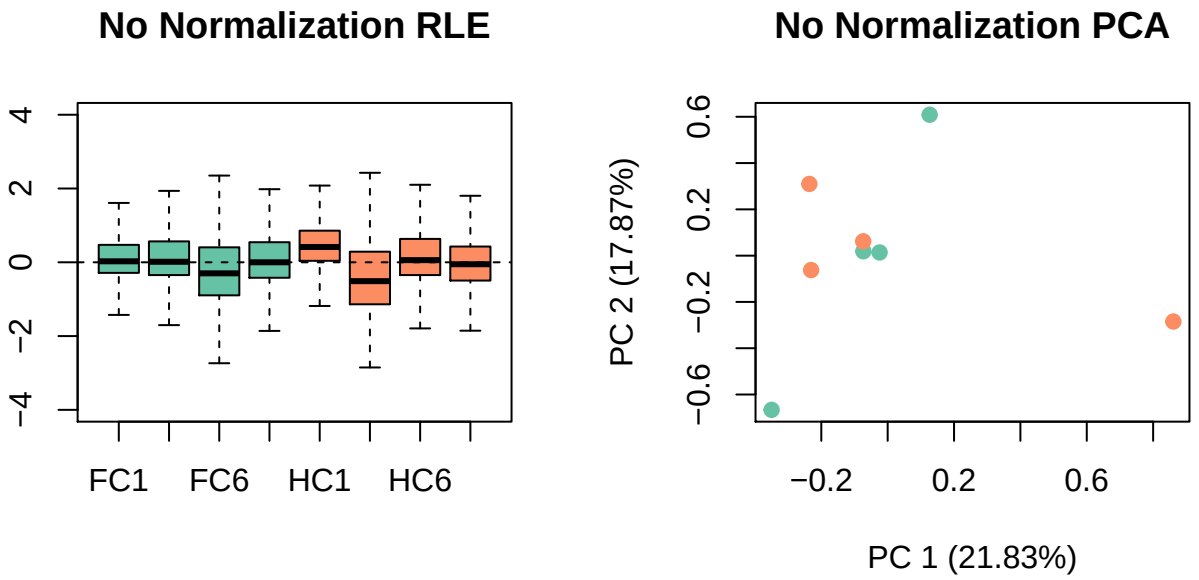
2.6 Normalization using RUV

In order to control for "unwanted variation", e.g., batch, library preparation, and other nuisance effects, the between-sample normalization method RUVs from the RUVSeq package can be utilized. Any normalization method based on total library counts is not appropriate for epigenetic sequencing experiments, as differences in total counts in the count matrix can be due to the treatment of interest.

```
library(RColorBrewer)
colors <- brewer.pal(3, "Set2")
set <- betweenLaneNormalization(count, which = "upper")
groups <- matrix(c(1:8), nrow = 2, byrow = T)
trt <- factor(c(rep("FC", 4), rep("HC", 4)))
```

The boxplots of relative log expression (RLE = log-ratio of read count to median read count across sample) and plots of principal components (PC) reveal a clear need for between-sample normalization.

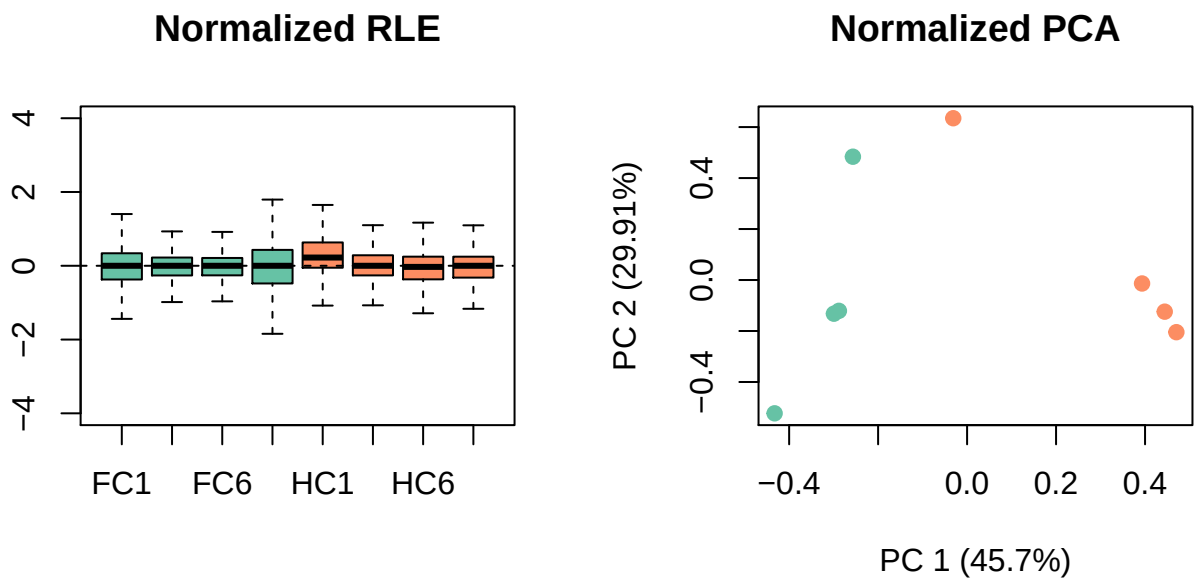
```
plotRLE(set, outline = FALSE, ylim = c(-4, 4),
        col = colors[trt], main = "No Normalization RLE")
plotPCA(set, col = colors[trt], main = "No Normalization PCA",
        labels = FALSE, pch = 19)
```



The parameter **k** dictates the number of factors of unwanted to variation to remove, in this case we use 4, but this is up for the user to determine. We can see in the PCA plot that after RUVs normalization the first 2 principal components seperate the two groups indicating that the treatment is the major source of variation.

```
k <- 4
s <- RUVSeq::RUVs(set, cIdx = rownames(set), scIdx = groups, k = k)

plotRLE(s$normalizedCounts, outline = FALSE, ylim = c(-4, 4),
        col = colors[trt], main = "Normalized RLE")
plotPCA(s$normalizedCounts, col = colors[trt], main = "Normalized PCA",
        labels = FALSE, pch = 19)
```



2.7 Testing for differential enrichment of regions

Now, we are ready to look for differentially enriched regions, using the negative binomial quasi-likelihood GLM approach implemented in edgeR (see the edgeR package vignette for details). This is done by considering a design matrix that includes both the covariates of interest (here, the treatment status) and the factors of unwanted variation.

```
design <- model.matrix(~0 + trt + s$W)
colnames(design) <- c(levels(trt), paste0("W", 1:k))

y <- DGEList(counts = count, group = trt)
y <- estimateDisp(y, design)

fit <- glmQLFit(y, design, robust = T)

con <- makeContrasts(FC - HC, levels = design)

qlf <- glmQLFTest(fit, contrast = con)
res <- topTags(qlf, n = Inf, p.value = 0.05)
head(res$table)
##                            logFC logCPM     F    PValue       FDR
## chr19:18964759-18965159 -22.79  7.281 37.89 8.224e-10 4.301e-06
## chr19:8990759-8991159   -20.73  6.991 35.09 3.407e-09 8.909e-06
## chr19:57212959-57213359 -14.33  6.759 28.33 1.076e-07 1.486e-04
## chr19:48117809-48118209 -13.27  6.552 28.23 1.136e-07 1.486e-04
## chr19:49429259-49429659 -12.71  6.279 27.43 1.707e-07 1.786e-04
## chr19:22916459-22916859 -11.03  6.133 25.29 5.142e-07 4.482e-04
dim(res$table)
## [1] 16  5
```