MeinteR: A framework to prioritize aberrant DNA methylation based on conformational and cis-regulatory element enrichment

9 September 2019

Contents

1 Overview

MeinteR (MEthylation INTERpretation) is an R package that identifies critical differentially methylated sites (DMS), based on the following hypothesis: Critical methylation-mediated changes are more likely to occur in genomic regions that are enriched in cis-acting regulatory elements than in "genomic deserts". With MeinteR we first calculate the abundance of co-localized elements, such as transcription factor binding sites, tentative splice sites, and other DNA features, such as G-quadruplexes and palindromes, that potentially lead to distinct DNA conformational features, and then we rank them with respect to their putative methylation impact. MeinteR supports the most widely-used, single-base resolution assays, including Illumina's BeadChip HumanMethylation27, HumanMethylation450 and MethylationEPIC arrays, whole genome, reduced representation and targeted bisulfite sequencing. The input data are either tabular files containing DMS coordinates of human genome (hg19 assembly) or array-based and sequencing data that can be directly imported from the Gene Expression Omnibus (GEO) repository. # Installation To download and install MeinteR and its dependencies, you need to install `devtools` and then run the following R commands:

```
devtools::install_github("andigoni/meinter")
```

Alternatively, download the binary installation file, unzip it to a folder e.g. `~/MeinteR` and run the following commands:

```
package.folder <- "~/MeinteR"
install_local(package.folder)
```

Upon successful installation of the package, load MeinteR and all its dependencies in the workspace using:

```
library(MeinteR)
```

2 Getting started

Before we start, we may carry out a couple of optional preparatory functions, as described below: ## Setting the working environment First, we set a title for our study by using the `nameStudy` function:

```
nameStudy(study.name="MyProj")
```

The `study.name` will be shown in the produced plots. Second, we set the analysis folder with read-write access rights e.g.:

```
project.dir <- "~/meinter_dir"
```

2.1 Formatting and importing datasets

2.1.1 Data format

MeinteR's core functions are applied on bed-formatted files containing the following columns:

Chr: Name of the chromosome (e.g. chr3)

Start: Start coordinate (hg19 assembly)

End: End coordinate (hg19 assembly)

Score: Difference of methylation levels ranging between -1 and 1

Strand: DNA strand. Either "." (=no strand) or "+" or "-"

The `Score` field of the input dataset may correspond to the `delta-beta` value of each DMS i.e. the difference of the methylation levels in two groups of data, for example normal/disease samples or pre-, post-treatment conditions of a sample set etc.

In the following, we demostrate the functionalities on a sample dataset assigned to variable `sample`. The `sample` dataset is automatically loaded in the workspace and can be used for demostration purposes.

```
head(sample)
```

The `sample` dataset contains the coordinates and `delta-beta` values for a set of 5,840 CpG sites. The column names are not consistent with formatting rules of the input data, so we need to reorder the misplaced columns using the `reorderBed` function:

```
re.sample = reorderBed(sample,1,2,3,5,4)
head(re.sample)
```

Columns 4 and 5 in the above example will change position. In addition, all columns will be renamed, appropriately. ### Import local data To work on your own data, load the tabular file by setting the column separator (`sep`) and heading (`header`) parameters appropriately. For example,

```
input.data <- read.csv(file.path(project.dir, "my_data.csv"), sep=",", header = T)
```

where `my_data.csv` in this example is a comma-delimited file including a header line. Additional columns in the input data will be ignored. MeinteR also provides a well-formatted dataset loaded in the workspace called `test.data` that contains 401 DMS with '|delta-beta|>=0.3'.

2.1.2 Import data from GEO

MeinteR supports analyses of GEO data series from microarray and next-generation sequencing platforms. In both cases, MeinteR will automatically fetch data series based on the accession number of the GEO data series. A data series usually contains moew than one samples analyzed by a unique or multiple platforms. MeinteR imports data from a data series that contains two groups of samples analysed by the same platform according to a user-defined annotation file. The annotation file is a comma-delimited file with two columns: Column `sample` lists the GSM accession numbers that are included in the data series and the `status` column lists the class of each sample. For example, to import sample data of the GSE37362 series we must create an annotation file that has the following form:

```
sample,status
GSM916781,WT
GSM916782,WT
GSM916783,WT
GSM916803,Mutant
GSM916804,Mutant
GSM916805,Mutant
```

The `status` column must list only two groups of samples e.g. WT/Mutant, pre-treatment/post-treatment etc. The analysis will be performed on the samples listed in the annotation file. Samples included in the data series, but not in the annotation file, will be ignored. MeinteR will automatically identify the platform and will fetch the corresponding sample data. In case of array-based data, the probes will be mapped to the hg19 genomic coordinates and a bed-formatted file will be generated conforming with the formatting rules of the input data. The delta-beta values will be automatically calculated by subtracting the mean beta values of the two groups. To import a GEO data series use the function `importGEO` as follows:

```
fp <- file.path(project.dir, "GSE37362_annotation.csv")
geo.data <- importGEO(gse.acc="GSE37362", annotation.file= fp)
```

where `gse.acc` is the unique GEO identifier of the data series and `annotation.file` is the local path to the annotation file.

2.1.3 Import `limma` differentially methylated probes

To increase compatibility with other software packages MeinteR includes an import function of the `Limma`-based differentially methylated probes. `Limma` is widely-used to build linear models for microarray data, includind DNA methylation. Using the `importLimma` function the highly ranked differentially methylated probes can be directly imported to the workspace and processed by the core functions. An example on the `minfi` data is shown below:

```
require(minfi)
require(minfiData)
require(limma)
library(MeinteR)
# Get example methylation data and transform to M-values
meth <- getMeth(MsetEx)
unmeth <- getUnmeth(MsetEx)
```

```
Mval <- log2((meth + 100)/(unmeth + 100))
group <- factor(pData(MsetEx)$Sample_Group)
#Build the design matrix
design <- model.matrix(~group)
#Fit a linear model
lFit <- lmFit(Mval,design)
#Compute Bayes statistics
lFit2 <- eBayes(lFit)
#Extract highly ranked probes
lTop <- topTable(lFit2,coef=2,num=200)
bed.data <- importLimma(lTop, sortBy="adj.P.Val")
pals <- findPals(bed.data)
#Find G-quadruplexes
quads <- findQuads(bed.data, offset = 50)
#Find overlaps with of probes located in promoters with JASPAR TFBS
tfbs <- findTFBS(bed.data)
#Feature weights
weights = list()
weights[["tfbs"]] = 1
weights[["pals"]] = 1
weights[["quads"]] = 1
#Mapping MeinteR arguments with function outputs
funList = list()
funList[["tfbs"]] = tfbs
funList[["pals"]] = pals
funList[["quads"]] = quads
#Calculate genomic index
index <- meinter(bed.data, funList, weights)
```

`topbed` is a data frame that complies with the formatting rules of the MeinteR input files. To export `topbed` data `importLimma` needs the `lTop` data frame containing empirical Bayes statistics of the highly ranked probes. These probes are matched to the annotation file of the `platform` that they came from and the `score` column of the `topbed` data frame is filled with the values set by the `sortBy` parameter.

2.2 Data filtering

A common preprocessing step is subsetting the initial dataset using filtering of the `score` values. For example:

```
#Select DMS with delta-beta values equal to 0
subsample.1 <- re.sample[re.sample$score == 0,]
#Select DMS with delta-beta values greater than or equal to 0.30
subsample.2 <- re.sample[re.sample$score >= 0.30,]
#Select DMS with absolute delta-beta values greater than or equal to 0.60
subsample.3 <- re.sample[abs(re.sample$score) >= 0.60,]
```

3 Core functions

MeinteR offers a set of core functions that investigate the presence of the following methylation-mediated regulatory features:

Transcription factors: Two functions are implemented that examine the effect of methylation on the binding affinity of a) human transcription factors, and b) conserved human/mouse/rat transcription factors.

Splice sites and alternative splicing events: Splicing regulation is analysed by two functions: a) A function that identifies potential 5' and 3' splice sites in the DMS proximal regions, and b) a function that searches for known alternative splicing events overlapping our DMS data.

Symmetry elements/Palindromes: With this function DMS are examined with respect to the presence of palindromic regions in a user-defined region centered at the DMS.

G-quadruplex structures: With this function the DMS are examined with respect to the presence of G-quadruplex structures.

DNA shapes: This function quantifies the effect of DNA methylation on four DNA shape conformations, aiming to identify potential changes on the DNA-protein interactions.

3.1 Detection of transcription factor binding sites

3.1.1 JASPAR transcription factor binding sites

`findTFBS` identifies potential binding sites of human transcription factors in a region of `2*offset` nucleotides centered at the DMS, where offset defines the sequence length on each side and `persim` sets the similarity threshold for considering a trascription factor as candidate for binding. The list of transcription factors, on both strands, is extracted from the JASPAR 2018 database (Khan *et al.*, 2018) and processed using `TFBSTools` (Tan and Lenhard, 2016). By default, MeinteR reserves the maximum number of available cores. Still, for large datasets `findTFBS` is time-consuming. In such case, consider shortening the list of target transcription factors using the `td.ID` parameter and/or descreasing the number of DMS to those overlapping with promoters or CpG islands, using the `target` parameter. For example, to identify a list of transcription factor binding sites on `subsample.3` run:

```
#Transcription factors of interest
tf.ID = c("MA0003.1", "MA0019.1", "MA0004.1", "MA0036.3", "MA0037.3")
tfbs <-findTFBS(bed.data=subsample.3,persim=0.8, offset=10, target="PROMOTER",
                up.tss=2000, down.tss=100, mcores = 2, tf.ID=tf.ID)
```

In the above example, `findTFBS` will return transcription factors that are inluded in the `tf.ID` list as well as the binding sequence that resembles at least 80% with the DMS expanded by 10bp on each side. The analysis is performed on DMS located in promoters (2000bp upstream and 100bp downstream TSS). `tfbs` is a list of two data frames. The first data frame `tfbs[[1]]` contains all the details of each transcription factor found at each DMS and `tfbs[[2]]` summarizes the number of transcription factors per DMS. `plotTF` function gets as input the `tfbs[[1]]` data frame and visualises the most frequent transcription factors that potentially bind to DMS regions.

```
plotTF(tfbs[[1]], topTF=10) #topTF:Number of most frequent transcription factors
```

`plotTF` returns a list of three plots (two bar charts and a scatter plot): The first bar chart overlays the total number of binding sites for each JASPAR transcription factor and the number of sequences containing at least one binding site (Figure 1). The second bar chart sums the number of binding sites per transcription factor class (Figure 2). Finally, the scatter plot combines the number of transcription factors per class that are identified in our dataset as compared with the JASPAR2018 number of transcription factors in each class (Figure 3).

3.1.2 Conserved Human/Mouse/Rat transcription factor binding sites

The function `findConservedTFBS` enables the detection of differentially methylated sites overlapping conserved transcription factor binding sites in the human/mouse/rat alignment. The score and threshold are computed using the Transfac Matrix Database (v7.0) available through UCSC Table Browser. The data are purely computational, and as such, not all binding sites listed here are biologically functional binding

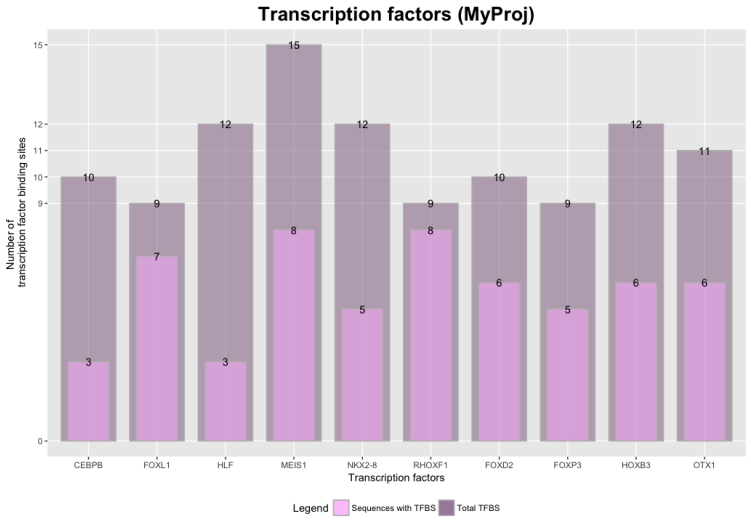

Figure 1: Most frequent transcription factors with putative binding sites on the DMS-containing sequences.

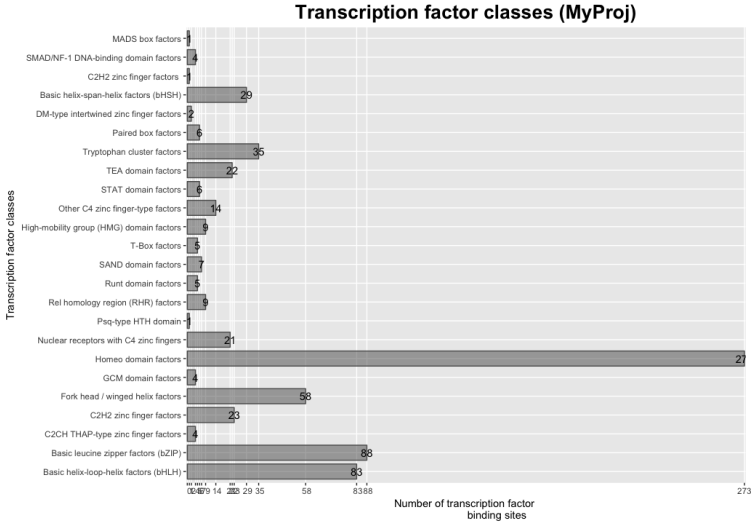

Figure 2: Classes of transcription factors with putative binding sites on the DMS-containing sequences.

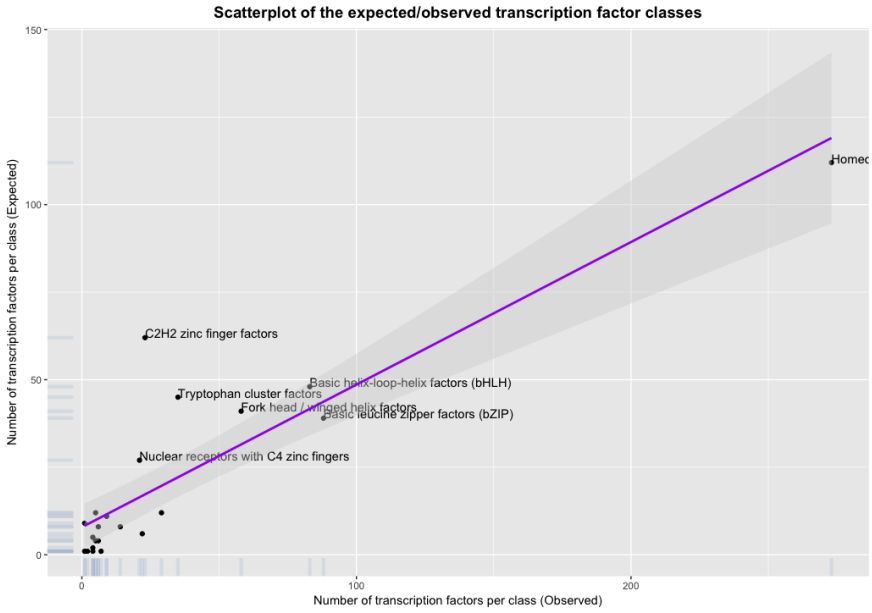

Figure 3: Scatter plot of the transcription factors in our dataset (Observed) vs. the total number of transcription factors in each class according to JASPAR108 database (Expected).

sites. `findConservedTFBS` will either fetch the transcription factor binding sites from UCSC or load a local copy of the file. To speed up the analysis it is advisable to download the `tfbsConsSites.txt.gz` file from http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/ and set the local path of this file to the `known.conserved.tfbs.file` argument of the `findConservedTFBS` function.

```
ctfbs <-findConservedTFBS(subsample.3, known.conserved.tfbs.file="~/tfbsConsSites.gz")
```

The function `findConservedTFBS` returns a list of three data frames. The first data frame contains the loci and names of the identified factors and the second lists the frequencies of each transcription factor, as well as the relative frequencies throughout the human genome. The latter data frame can be visualised by a scatterplot (Figure 4), using the function `scatterConsTF`:

```
scatterConsTF(ctfbs[[2]])
```

The third data frame lists the number of conserved transcription factors that overlap with each DMS. ## Detection of splice sites and alternative splicing events ### Detection of putative splice sites To detect the presence of 5' and 3' splice sites, MeinteR employs the Shapiro and Senapathy (S&S) method (Shapiro and Senapathy, 1987) that is based position-specific weight matrices. The function `findSpliceSites` implements the S&S method and employs the scanning method that is implemented in `TFBSTools`. For example, to detect putative splice sites in the 10nt regions centered at the DMS (80% similarity threshold), use the following command:

```
ss <- findSpliceSites(bed.data=subsample.3, persim=0.8, offset= 10)
```

`ss` is a list of two data frames, one containing the number of splice sites per sequence, and the second is a data frame containing the relative coordinates, type and score of all splice sites, as shown below:

```
head(ss[[1]])
head(ss[[2]])
```

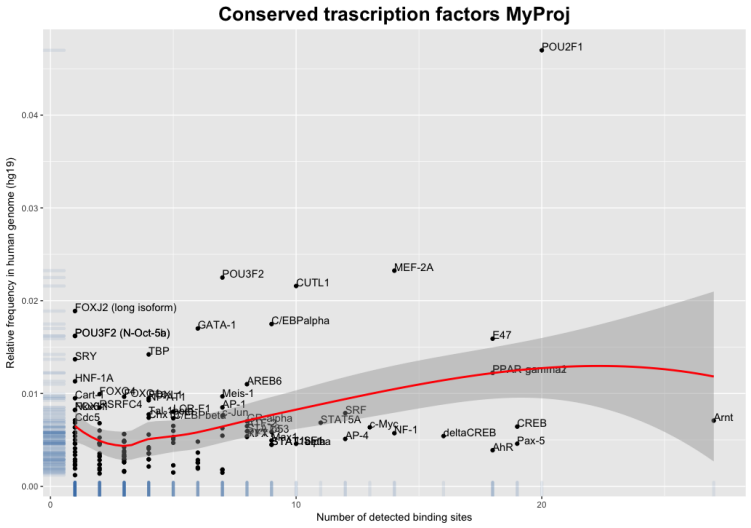

Figure 4: Scatterplot of the conserved transcription factors and their relative frequency in the reference genomes.

3.1.3 Map data to UCSC alternative splicing events

`findAltSplicing` maps DMS sequences to alternative splicing events (UCSC Alt events track) and exports their frequency and type in tabular and graphical representations. An example on the `subsample.3` dataset is the following:

```
altss <- findAltSplicing(subsample.3)
```

`findAltSplicing` fetches known alternative splicing events from UCSC Table Browser (needs internet connection) and exports three data frames: 1) A summary report, 2) a detailed report listing the number and frequencies of DMS co-localized with the alternative splicing events, and 3) an overlayed bar plot summarizing the distribution of the alternative splicing events in the reference genome and the input data (Figure 5).

```
head(altss[[1]])
head(altss[[2]])
head(altss[[3]])
altss[[4]] #Plot alternative splicing events
```

3.2 Symmetry elements - Palindromes

MeinteR enables the detection of palindromes in the genomic region neighboring DMS. Function `findPals` uses the `Biostrings` package to identify palindromic sequences in a user-defined region centered at DMS. This procedure is iteratively applied on the bed-formatted dataset and the function returns a DNAString object with the palidromes included in each sequence, together with the total number of DMS inside and outside palindromes.

```
pals <- findPals(bed.data=subsample.3, offset=10, min.arm=5, max.loop=5, max.mismatch=1)
```

In this example, `findPals` will search for palindromes in each sequence of the `subsample.3` dataset. Each sequence contains 10 nucleotides (`offset`) adjacent to DMS and palindromes with at least 5nt in each arm (`min.arm`), and 5nt maximum number of nucleotides inbetween of the two arms (`max.loop`) are identified, allowing 1 mismatch (`max.mismatch`) at most between arms.

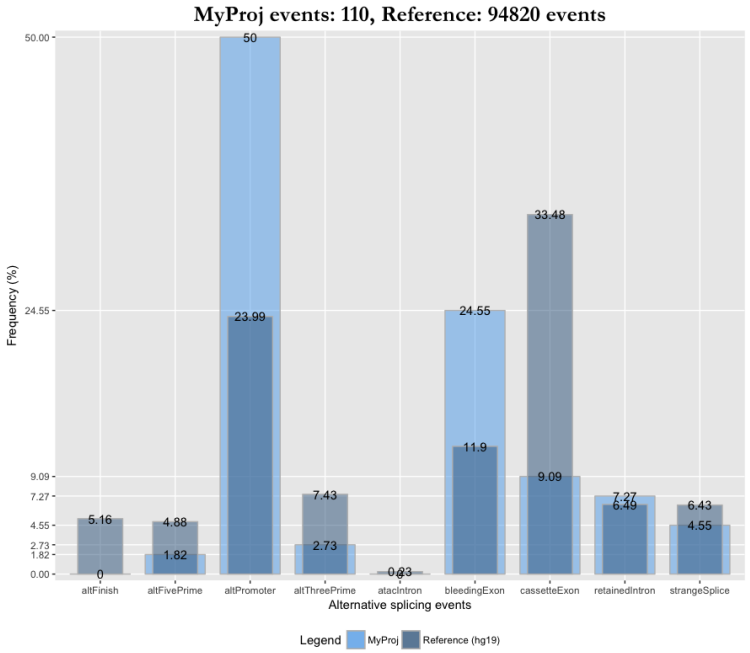

Figure 5: Distribution of methylation data to different alternative splicing events.

```
head(pals[[1]],n=1)
pals[[2]] # On/Off DMS palindromes
head(pals[[3]]) #Number of palindromes per DMS
```

3.3 G-quadruplex structures

G-quadruplexes (G4) are DNA structures that are involved in various biological processes. The formation of G4 has been also studied with respect to DNA methylation showing a strong association in several studies (Halder *et al.*, 2010),(Stevens and Kennedy, 2017). MeinteR provides the `findQuads` function that detects G4 structures based on the `pqsfinder` method (Hon *et al.*, 2017). `findQuads` gets a valid methylation dataset and exports the identified G4, their relative position in the sequences of `offset` nucleotides adjacent to DMS and the abundance per sequence. An example execution on the `subsample.3` dataset is shown below:

```
#Detect G4s in the 100nt region flanking DMS
quads <- findQuads(bed.data=subsample.3, offset=100)
quads[[1]] # G4 locus information for each sequence
quads[[2]] # G4 on/neighboring input data
quads[[3]] # Number of G4 per sequence
```

The arguments of `findQuads` are bed-formatted data together with the sequence offset and the function exports a list of three data frames.

3.4 Other DNA structures

`findShapes` is a function that enables the detection of DNA shape alterations caused by DNA methylation. Given a set of DMS, `findShapes` determines the conformational changes, based on the `methyl-DNAshape` (Rao *et al.*, 2018). The function gets as input a bed-formatted dataset `bed.data` and the `offset`. The output

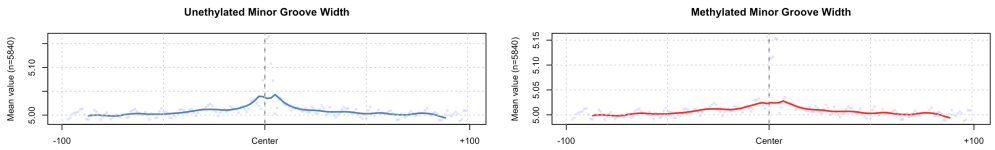

Figure 6: Changes introduced to the minor groove width in the `sample` dataset.

is a list of four vectors containing the p-value of the helix twist, minor groove width, propeller twist, and roll in the unmethylated and methylated context. (Chiu *et al.*, 2016).

```
#Detect DNA shapes in the 100nt region flanking DMS in the re.sample data
shapes <- findShapes(bed.data=subsample.3, offset=100)
```

In addition, `findShapes` function returns a multi-panel plot and four data frames with the distribution of each DNA feature in the unmethylated and methylated context (more on the function's help page).

Figure 6 illustrates the mean values of the minor groove width in case of unmethylated and methylated DMS of the `subsample.3` dataset. Similar plots are generated for the other DNA shape features. For CpGs that are differentially methylated, `findShapes` gives insights on how DNA methylation induce conformational changes in a DNA sequence.

4 Genomic signatures and ranking

In the final step, MeinteR builds genomic signatures of the DMS using the `meinter` function. `meinter` gets as input a) the results (all of subset) of functions `findAltSplicing`, `findConservedTFBS`, `findPals`, `findQuads`, `findShapes`, `findSpliceSites` and `findTFBS`, and b) a list of the relative weights. Then it builds a matrix with the weighted frequencies of each genomic feature per DMS and calculates the genomic index of each position. For example:

```
#Calculate genomic index
weights = list()
weights[["spls"]] = 1
weights[["ctfbs"]] = 2
weights[["tfbs"]] = 2
weights[["pals"]] = 1
weights[["quads"]] = 2
weights[["shapes"]] = 1
funList = list()
funList[["spls"]] = ss
funList[["altss"]] = altss
funList[["tfbs"]] = tfbs
funList[["ctfbs"]] = ctfbs
funList[["pals"]] = pals
funList[["quads"]] = quads
funList[["shapes"]] = shapes
index <- meinter(subsample.3, funList, weights)
head(index) # Highly ranked DMS
```

`index` contains the genomic index of each DMS in descending order.

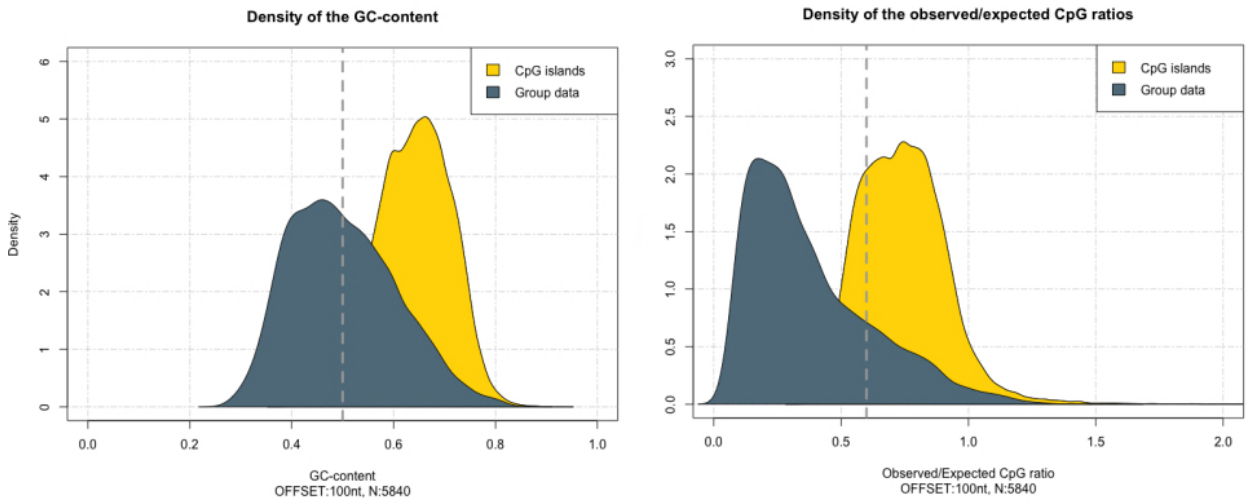

Figure 7: Density plots of the GC-content and the `Observed/Expected` CpG ratio of our data (Group data) compared with the CpG islands of the human genome (CpG islands). N is the number of the examined sequences. OFFSET is the expansion of the region length on each side of the methylation site.

5 Supplementary functions

5.1 GC/CpG content

The input methylation data can be examined with respect to the presence of neighboring CpG-dense regions, CpG islands, using the `plotCpG` function. `plotCpG` generates density plots of: a) GC-content i.e. the sum of the C and G occurrences in the DNA sequence divided by the total sequence length, and b) the `Observed/Expected` CpG ratio, where `observed` corresponds to the number of CpGs i.e. the number of CpG dinucleotides in the DNA sequence, and `Expected` is the number of Cs multiplied by the number of Gs divided by the total sequence length (Figure 7). The function also returns a data frame with the corresponding GC-content and observed/expected ratios of the input data.

```
dev.off() #close any open plot device
res <- plotCpG(bed.data=re.sample, offset=100)
```

`plotCpG` gets as input the bed-formatted dataset and the `offset` of the sequence centered to the DMS.

5.2 Plot distribution of the methylation levels

The density plot of the input data can be visualized using the `plotBeta` function. `plotBeta` gets the `score` column of the input data and plots the density using a Gaussian kernel. The `score` column can be either beta or delta-beta values.

```
plotBeta(bed.data=re.sample)
```

6 Working with real data

6.1 Example with Illumina HumanMethylation450K data (GSE37362)

In the following example, we demostrate the functionalities of MeinteR on a real dataset. We use data from the GSE37362 data series that contains HumanMethylation450K methylation profiles of 31 diffuse large B-cell lymphoma, classified in two groups depending on whether TET2 is mutated. Briefly, we will use the `importGEO` function to fetch methylation data from GEO and to calculate the mean beta-values for each group and the delta-beta values for each probe. `importGEO` exports a valid bed-formatted data frame, including all probes transformed into the corresponding hg19 genomic coordinate. In case of unresponsive attempts to download GEO data series or download with warnings, try again to fetch data with the same scripts.

```
project.dir <- "~/GSE37362"
setwd(project.dir)
library(MeinteR)
gse.accession <- "GSE37362"
annotation.file <- file.path(project.dir,"GSE37362_annotation.csv")
geo.data <- importGEO(gse.acc=gse.accession, annotation.file=annotation.file)
```

`geo.data[[1]]` is a data frame that contains the bed-formatted dataset. We use the `reorderBed` function to transform `geo.data[[1]]` and remove rows with undefined values.

```
# Re-order columns and omit rows with empty cells
bed.data<-na.omit(reorderBed(geo.data[[1]],3,4,5,2))
```

`geo.data` includes also two data frames with the beta values of each sample and the annotation file (refer to the function's help page).

```
#Select probes with delta-beta < -0.2
sub.data <- subset(bed.data, bed.data$score < -0.2) # 713 probes
quads <- findQuads(sub.data, offset=100)
pals <- findPals(sub.data, offset=100)
altspl <- findAltSplicing(sub.data)
tfbs <- findTFBS(sub.data, target="PROMOTER", up.tss=5000, down.tss=100)
ss <- findSpliceSites(sub.data)
shapes <- findShapes(sub.data, offset=100)
```

6.2 Example with next-generation sequencing data

Working with public sequencing data requires additional pre-processing, since the types of data may vary. MeinteR provides the `loadSeqGEO` function that transforms bisulfite-sequenced data from WGBS, RRBS and targeted sequencing experiments into valid input data. `loadSeqGEO` calculates methylation levels and filters out low-covered CpGs, using a user-defined coverage threshold. `loadSeqGEO` has to be applied iteratively to all samples and subsequently merge data, intersect and calculate differentially methylation level for each CpG site between the two sample groups, according to each analysis scenario. An example usage of the `loadSeqGEO` for the GSE69272 data series is shown in the following:

```
#Transform all the bed.gz files in the ~/GSE69272_RAW folder
files <- list.files(path="~/GSE69272_RAW", pattern="*.bed.gz",
                    full.names=T, recursive=FALSE)
all.samples <- lapply(files, function(samples) {
    loadSeqGEO(file.path=samples, cov=30, chroms="chr19")
})
```

`all.samples` is a list containing the methylation values of the CpGs in chromosome 19 that are covered by at least 30 reads. The list includes all the samples (bed.gz files) that have been downloaded in the `~/GSE69272_RAW` folder.

6.3 Demo run

The basic steps of the pipeline for the `sample` dataset can be summed up in the following commands:

```
library(MeinteR)
rm(list = ls())
re.sample <- reorderBed(sample, 1, 2, 3, 5, 4)
bed.data <- re.sample[re.sample$score >= 0.50,]
altSS <- findAltSplicing(bed.data)
ss <- findSpliceSites(bed.data, persim = 0.8, offset = 10)
pals <- findPals(bed.data)
quads <- findQuads(bed.data, offset = 50)
tfbs <-
  findTFBS(
    bed.data,
    target = "all",
    tf.ID = c("MA0107.1", "MA0098", "MA115.1", "MA0131.2")
  )
#Find overlaps with conserved transcription factors. ~71MB gzipped file
#will be downloaded if local path is not set.
ctfbs <- findConservedTFBS(bed.data)
shapes <- findShapes(bed.data)
weights = list()
weights[["spls"]] = 1
weights[["ctfbs"]] = 1
weights[["tfbs"]] = 1
weights[["pals"]] = 1
weights[["quads"]] = 1
weights[["shapes"]] = 1
funList = list()
funList[["spls"]] = ss
funList[["altss"]] = altSS
funList[["tfbs"]] = tfbs
funList[["ctfbs"]] = ctfbs
funList[["pals"]] = pals
funList[["quads"]] = quads
funList[["shapes"]] = shapes
index <- meinter(re.sample, funList, weights)
```

References

Chiu,T.-P. *et al.* (2016) DNAshapeR: An R/Bioconductor package for DNA shape prediction and feature encoding. *Bioinformatics*, **32**, 1211–1213.

Halder,R. *et al.* (2010) Guanine quadruplex DNA structure restricts methylation of CpG dinucleotides genome-wide. *Molecular BioSystems*, **6**, 2439.

Hon,J. *et al.* (2017) Pqsfinder: An exhaustive and imperfection-tolerant search tool for potential quadruplex-

forming sequences in R. *Bioinformatics*, **33**, 3373–3379.

Khan,A. *et al.* (2018) JASPAR 2018: Update of the open-access database of transcription factor binding profiles and its web framework. *Nucleic Acids Research*, **46**, D260–D266.

Rao,S. *et al.* (2018) Systematic prediction of DNA shape changes due to CpG methylation explains epigenetic effects on proteinDNA binding. *Epigenetics & Chromatin*, **11**.

Shapiro,M.B. and Senapathy,P. (1987) RNA splice junctions of different classes of eukaryotes: Sequence statistics and functional implications in gene expression. *Nucleic Acids Res.*, **15**, 7155–7174.

Stevens,A.J. and Kennedy,M.A. (2017) Methylated Cytosine Maintains G-Quadruplex Structures during Polymerase Chain Reaction and Contributes to Allelic Dropout. *Biochemistry*, **56**, 3691–3698.

Tan,G. and Lenhard,B. (2016) TFBSTools: An R/bioconductor package for transcription factor binding site analysis. *Bioinformatics*, **32**, 1555–1556.