Package ‘MMDiff’

March 26, 2015

Type Package

Title Statistical Testing for ChIP-Seq data sets

Version 1.7.0

Date 2012-03-22

Author Gabriele Schweikert

Maintainer Gabriele Schweikert <g.schweikert@ed.ac.uk>

Description This package detects statistically significant difference between read enrichment profiles in different ChIP-Seq samples. To take advantage of shape differences it uses Kernel methods (Maximum Mean Discrepancy, MMD).

License Artistic-2.0

Imports GenomicRanges,IRanges,Biobase

Depends R (>= 2.14.0),GenomicRanges,parallel,DiffBind,GMD,Rsamtools

Suggests MMDiffBamSubset

biocViews ChIPSeq, MultipleComparison

NeedsCompilation no

R topics documented:

MMDiff-package .................................................. 2
Cfp1Dists .................................................. 3
Cfp1Profiles .................................................. 4
compHistDists .................................................. 5
detPeakPvals .................................................. 8
findOutliers .................................................. 9
getNormFactors ............................................... 10
getPeakProfiles ............................................... 12
plotHistDists ............................................... 15
plotPeak .................................................. 16

Index 18
Description

This package detects statistically significant difference between read enrichment profiles in different samples. To take advantage of shape differences it uses Kernel methods (Maximum Mean Discrepancy, MMD, [1]).

Details

The starting point for this package is a DBA object created with the package DiffBind [2]. Sample specific peak profiles (histograms) can then be generated for a specified set of peaks. Rsamtools are used to load reads from bam files, strand shifts are corrected and histograms are computed for each peak and sample. Differences between samples at each peak are assessed by computing distances between the corresponding histograms in terms of Maximum Mean Discrepancy (MMD) or Generalized Minimum distance (GMD) [3], taking structural information into account [1]. Empirical p-values can be determined for a comparison of two sets of samples (e.g. control samples vs. treatment samples). Examples are provided using partial data from [3].

Function list:

- getPeakProfiles: Add histograms (binned read enrichment profiles) to an existing DBA object
- findOutliers: Find peaks with extreme count values
- getNormFactors: Determine normalisation factors between samples
- compHistDists: For each peak, compute distances of histograms between pairs of ChIP-Seq data sets (using Maximum Mean Discrepancy)
- detPeakPvals: Determine p-values for each peak comparing two groups of data sets
- plotHistDists: For each peak plot computed distances as a function of total counts, show peaks which are significantly different
- plotPeak: Plot read enrichment profiles for a set of samples at a given peak

Author(s)

Gabriele Schweikert

Maintainer: Gabriele Schweikert <G.Schweikert@ed.ac.uk>

References

Cfp1Dists


**Description**

The object contains a small subset of the CIP-Seq data sets generated to assess the link between the histone modification states of H3K4me3 with respect to the mediator proteins Cfp1 [1]. The data is available as part of ArrayExpress Experiment E-ERAD-79. For more details see the MMDiffBamSubset data package.

**Usage**

data(Cfp1Dists)

**Format**

A DBA object containing the additional component MD. A DBA object is an S3 object (class "DBA"), which is introduced with the DiffBind package and more information on this class can be found in the Diffbind vignette.

The added field MD is a list containing the following elements:
- RawTotalCounts: matrix of total counts per peak and sample (nPeaks x nSamples)
- PeakRawHists: list of length nPeaks, containing for each Peak a Matrix of histograms (nSamples x nbins). Note, as Peaks can vary in length, nbins may be different for each Peak.
- NormFactors: Normalization factors
- NormTotalCounts: Normalized total counts
- DISTS: containing distances computed for ‘MMD’, ‘GMD’ and ‘Pearson’. Each being a (nPeaks x 3) matrix containing in the first column the distances between pairs of peaks in the WT vs Null samples, in the second WT vs Resc and in the third Null vs Resc

**Source**

See the MMDiffBamSubset data package for more details.

**References**

See Also
dba, summary.DBA

Examples

# The object has been generated using the following commands:
# STEP1: load peak profiles and normalize

data(Cfp1Profiles)

Cfp1Norm <- getNormFactors(Cfp1Profiles)

# STEP2: compute distances between histograms

Cfp1Dists <- compHistDists(Cfp1Norm, method='MMD', NormMethod='DESeq')
Cfp1Dists <- compHistDists(Cfp1Dists, method='GMD', NormMethod='DESeq')
Cfp1Dists <- compHistDists(Cfp1Dists, method='Pearson', NormMethod='DESeq')

Cfp1Profiles

Example DBA object used to illustrate usage of the MMDiff package

Description

The object contains a small subset of the CIP-Seq data sets generated to assess the link between the histone modification states of H3K4me3 with respect to the mediator proteins Cfp1 in mouse [1]. The data is available as part of ArrayExpress Experiment E-ERAD-79. For more details see the MMDiffBamSubset data package.

Usage
data(Cfp1Profiles)

Format

A DBA object containing the additional component MD. A DBA object is an S3 object (class "DBA"), which is introduced with the DiffBind package and more information on this class can be found in the Diffbind vignette.

The added field MD is a list containing the following elements:
- RawTotalCounts: matrix of total counts per peak and sample (nPeaks x nSamples)
- PeakRawHists: list of length nPeaks, containing for each Peak a Matrix of histograms (nSamples x nbins). Note, as Peaks can vary in length, nbins may be different for each Peak.

Source

See the MMDiffBamSubset data package for more details.
compHistDists

References


See Also
dba, summary.DBA

Examples

# The object has been generated using the following commands:

### STEP1: create DBA object:

library('MMDiffBamSubset')
oldwd <- setwd(system.file("extdata", package="MMDiffBamSubset"))

Cfp1 <- dba(sampleSheet="Cfp1.csv", minOverlap=3,
  config = data.frame(RunParallel=FALSE))

### STEP2: compute histograms from bam files

bin.length <- 50
Peaks <- dba.peakset(Cfp1,bRetrieve=TRUE)
Peaks <- Peaks[1:1000]
Cfp1Profiles <- getPeakProfiles(Cfp1,Peaks,bin.length=bin.length,
  save.files=FALSE,run.parallel=FALSE)
setwd(oldwd)

compHistDists  Compute distances between pairs of histograms

Description

This function computes for each peak pairwise distances between histograms according to the specified method, currently Maximum Mean Discrepancy (MMD), Generalized Minimum Distance (GMD) and simple Pearson correlation (Pearson) are implemented.

Usage

```r
compHistDists(DBA, method = 'MMD', CompIDs=NULL, Usefiltered = TRUE,
  PeakIDs = NULL, NormMethod = 'DESeq',
  overwrite = FALSE, HistField = 'PeakRawHists',
  run.parallel = TRUE, verbose = 2,
  save.file = TRUE, out.dir='.', sigma=NULL)
```
**Arguments**

- **DBA**
  DBA object, after running `getPeakProfiles`. Specifically, it uses the element MD, which contains a list of histogram matrices. (see the `getPeakProfiles` documentation for more information about this data type.)

- **method**
  specify what method should be used to determine distances between histograms, could be 'MMD' [1], 'GMD' [2] or simple 'Pearson' correlation

- **CompIDs**
  2 x nComps matrix, specifying sample ids of pairwise comparisons

- **Usefiltered**
  If TRUE, only peaks that have passed the filter to detect Outliers are considered. `findOutlier()` must be run first, otherwise all peaks are used

- **PeakIDs**
  Specify a subset of peaks for which distances should be completed

- **NormMethod**
  specify which normalization method should be used, currently only the 'DESeq' method [3] is implemented. Note, that unless NormMethod=NULL, getNormFactors has to be called first.

- **overWrite**
  if TRUE, overwrites earlier computed distances.

- **HistField**
  name of element in MD that is used to determine distances. This element should again be a list of nPeaks peaks, each containing a matrix of histograms (nSamples x nbins). It can be generated by running `getPeakProfiles`. Note, nbins may vary between peaks, if they have different length.

- **run.parallel**
  distribute over available CPUs

- **verbose**
  for debugging, set to 3 for some extra output

- **save.file**
  if TRUE, DBA objects are saved

- **out.dir**
  directory for saving output files

- **sigma**
  parameter controlling the Kernel size

**Value**

DBA object, with additional list element DISTS added to MD. DISTS again contains a list element named according to method applied (e.g. MMD). This element is a matrix (nPeaks x nComps) containing all pairwise distances.

**Author(s)**

Gabriele schweikert

**References**


See Also

getPeakProfiles, findOutliers, getNormFactors, detPeakPvals, plotHistDists, plotPeak

Examples

# load DBA objects with peak profiles
data(Cfp1Profiles)

# get normalization factors
Cfp1Norm <- getNormFactors(Cfp1Profiles)

# get all pairwise distances for the samples WT, Null and Resc i.e. WT
# vs Null, WT vs Resc and WT vs Resc: Recommended is the method 'MMD'
# [1], however, this may take a little while. Here, we compute the GMD
# distance instead [2].
Cfp1Dists <- compHistDists(Cfp1Norm, method = 'GMD',
                           NormMethod = 'DESeq')

# You can also specify, which pairwise distances you are interested in,
# e.g.:
CompIDs <- cbind(c("WT.AB2", "Null.AB2"),
                 c("WT.AB2", "Resc.AB2"),
                 c("Null.AB2", "Resc.AB2"))

Cfp1Dists2 <- compHistDists(Cfp1Norm, method='GMD', CompIDs=CompIDs,
                           NormMethod='DESeq')

# To view pairwise distances you can use the function plotHistDists. For
# example, treating WT and Resc as control replicates and Null as a
# treatment group, you can contrast the 'within-group' distances with
# 'between-group' distances:

group1 <- c("WT.AB2","Resc.AB2")
group2 <- c("Null.AB2")

plotHistDists(Cfp1Dists, group1=group1, group2=group2, method='GMD')

#see detPeakPvals to determine which peaks are significantly different
#between the two groups.
**detPeakPvals**

*Compute p-values for each peak based on distances between histograms*

**Description**

Compute p-values for each peak based on distances between histograms, contrasting group1 (e.g. control samples) with group2 (e.g. treatment samples). To estimate within group distances and between group distances peaks are pooled according to their mean (normalized) total counts. p-values are adjusted for multiple testing using the method by Benjamini & Hochberg (1995).

**Usage**

```r
detPeakPvals(dba, method = "MMD", group1, group2,
             name1 = "g1", name2 = "g2", Usefiltered = TRUE,
             PeakIDs = NULL,
             quantprobs = seq(0, 1, 0.05),
             fieldName = "NormTotalCounts", bNormWidth = FALSE,
             bSampleMean = FALSE, overwrite = FALSE)
```

**Arguments**

- **DBA**
  - **DBA object**, after running getPeakProfiles and compHistDists.
- **method**
  - which distance method should be used. (can be 'MMD', 'GMD' or 'Pearson')
- **group1**
  - sample ids of control group
- **group2**
  - sample ids of treatment group
- **name1**
  - name of control group
- **name2**
  - name of treatment group
- **Usefiltered**
  - If TRUE, only peaks that have passed the filter to detect Outliers are used. findOutlier must be run first, otherwise all peaks are used
- **PeakIDs**
  - specify a subset of peaks which should be used for pooling (for example if outliers with extreme counts should be excluded)
- **quantprobs**
  - numeric vector of probabilities with values in [0,1], used to specify which peaks are pooled together to estimate variances.
- **fieldName**
  - name of list element in DBA$MD that is used for pooling of peaks. (e.g. NormTotalCounts or RawTotalCounts)
- **bNormWidth**
  - logical indicating if counts should be normalized by peak width
- **bSampleMean**
  - If true counts are averaged across all samples. Otherwise means are computed for each group separately.
- **overwrite**
  - if TRUE, previous computed p-values are overwritten
findOutliers

Value

DBA object, with additional element Pvals added to MD. Pvals again contains a list element named according to method applied (MMD), e.g. DBASMDSPvals$MMD This element is a matrix (nPeaks x ncomps) containing p-values for each peak and given comparison (group1 vs. group2). New comparisons (i.e. re-running detPeakPvals with different groups) are appended to the matrix.

Author(s)

Gabriele Schweikert

See Also

gربحProfiles, getNormFactors, compHistDists, plotHistDists, plotPeak

Examples

# load DBA objects with peak profiles and pairwise distances
data(Cfp1Dists)

# specify controll and treatment groups:
group1 <- c("WT.AB2", "Resc.AB2")
group2 <- c("Null.AB2")

# determine empirical p-values:
Cfp1Pvals <- detPeakPvals(Cfp1Dists, group1=group1, group2=group2,
                          name1='WT/Resc', name2='Null')

# to plot distances and peaks which are significantly different use the
# plotHistDists function:
plotHistDists(Cfp1Pvals, group1=group1, group2=group2)

findOutliers

Find peaks with extreme count values

Description

findOutliers uses the function boxplot to determine outlier peaks with extreme high total counts in each sample.

Usage

findOutliers(DBA, range = 20, draw.on=TRUE)
Arguments

**DBA**
DBA object, after running getPeakProfiles.

**range**
This parameter specifies what is classified as an outlier. Non-outliers extend to the most extreme data point which is no more than 'range' times the inter-quartile range from the box in boxplot. (In the function boxplot, range determines how far the plot whiskers extend out from the box.

**draw.on**
If TRUE, histograms and box plots are plotted for quality control.

Value

DBA object, with additional list element Filter added to MD.

Author(s)

Gabriele Schweikert

See Also

getPeakProfiles, boxplot

Examples

data(Cfp1Profiles)
Cfp1 <- findOutliers(Cfp1Profiles, range=20)
Filter <- Cfp1$MD$Filter

Cfp1Strict <- findOutliers(Cfp1Profiles, range=10)
OutlierPeakIDs <- Cfp1Strict$MD$Filter$HighPeakIds

---

getNormFactors  
**Determine Normalisation factors**

Description

Determine normalisation factors for a specified set of samples. Potentially only a subset of the peaks can be used to determine normalisation factors. The determined factors can be accessed with DBA$MD$NormFactors. Normalised total counts are additionally computed and stored at DBA$MD$NormTotalCounts.

Usage

getNormFactors(DBA, method = "DESeq", SampleIDs = NULL, Usefiltered = TRUE, PeakIDs = NULL, overWrite = FALSE)
getNormFactors

Arguments

- **DBA**  
  DBA object after running `getPeakProfiles`.
- **method**  
  currently only the DESeq normalisation method is implemented [1].
- **SampleIDs**  
  State which samples should be normalised; if NULL all are used.
- **Usefiltered**  
  If TRUE, only peaks that have passed the filter to detect Outliers are used.  
  `findOutlier()` must be run first, otherwise all peaks are used.
- **PeakIDs**  
  Specify a subset of peaks to be used to determine normalisation factors;  
  If NULL all peaks are used.
- **overwrite**  
  If TRUE, previous computed NormFactors and NormTotalCounts are overwritten

Value

DBA object, with additional list elements NormFactors and NormTotalCounts appended to MD.  
Note, that if you call `getNormFactors` several times with different parameters, you can have more  
than one set of normalisation factors appended. However, NormTotalCounts will be overwritten  
unless specified otherwise.

Author(s)

Gabriele Schweikert

References

Genome Biology, 11 (10): R106

See Also

- `getPeakProfiles`, `plotPeak`, `findOutliers`

Examples

```r
# load DBA objects with peak profiles
data(Cfp1Profiles)
Cfp1Norm <- getNormFactors(Cfp1Profiles)
Cfp1Norm$MD$NormFactors

# compare total counts before and after normalisation:
boxplot(Cfp1Norm$MD$RawTotalCounts[,1:3], ylim=c(0,2000))
boxplot(Cfp1Norm$MD$NormTotalCounts[,1:3], ylim=c(0,2000))

# compare individual peak profiles before and after normalisation,
# using plotPeak, e.g.:
plotPeak(Cfp1Norm, Peak.id=20, NormMethod = NULL)
```
getPeakProfiles

Add histograms (binned read enrichment profiles) to an existing DBA object

Description

This function is a wrapper that collects all 5' starting positions of mapped short reads in the bam.files that match to one of nPeaks regions defined by Peaks. It corrects for strand shifts and builds for each Peak a (nSamples x nBins) matrix containing histograms for each sample. Additionally computes total counts per peak and sample.

Usage

getPeakProfiles(DBA, Peaks, bin.length = 20, keep.extra = FALSE, draw.on = TRUE, save.files = FALSE, use.old = TRUE, out.dir = ".", run.parallel=TRUE)

Arguments

DBA
DBA object, which can be generated with the dba function of the DiffBind package. A standard DBA object is an S3 object (class "DBA"), which is introduced with the DiffBind package and more information on this class can be found in the Diffbind vignette. This function uses the following elements of the DBA object: chrmap, samples$bamReads, samples$bamControl

Peaks
GRanges Object containing genomic coordinates of the regions of interests.

bin.length
number of base pairs that are summarised per bin.

keep.extra
if TRUE extra information is kept, e.g. RawHists, which Counts.p and Counts.n corresponding to histograms on forward and reverse strand before strand shift is applied.

draw.on
if TRUE, strand shift plots are created for each data sample. This can be used for quality control (see details below)
getPeakProfiles

save.files  if TRUE, DBA objects are saved for each data sample
use.old    if TRUE, available files in out.dir are loaded
out.dir    directory for saving output files
run.parallel  distribute over available CPUs

Details

This function uses as a starting point a DBA object (see the DiffBind package for more details). The path to the bam files is stored in the samples$bamReads element of the DBA object. In addition, in samples$bamControl, bam files for the control samples should be specified. These files are accessed using Rsamtools to collect all short reads that match to regions defined by Peaks. 5’ Positions of reads are returned for reads mapping to positive and negative strand respectively. To correct for the strand shift, only peaks are selected whose total number of reads mapping to each strand is in the 9th decile. If draw.on=TRUE, a plot is generated for each bam file (all bamRead and bamControl files), showing smoothscatter plots of total number of reads mapping to the peaks on forward vs reverse strand. This can be used as a quality control (Points should lie on the diagonal). The peaks used to determine the strand shift are shown in red. For each of the selected peaks the shift between forward and reverse strand is determined using the cross-correlation function ccf. If draw.on=TRUE, histograms are plotted for each bam file, showing the distribution of shifts. The median of shifts is used to correct all reads mapping to any peak in the respective bam file. (Note, the shifts can vary between samples i.e. different bam files.)

Value

DBA object, with additional component MD, which is a list containing

PeakRawHists  list of length nPeaks, containing for each Peak a Matrix of histograms (nSamples x nbins). Note, as Peaks can vary in length, nbins may be different for each Peak
RawTotalCounts  matrix of total counts per peak and sample (nPeaks x nSamples)
RawHists  Only provided if keep.extra=TRUE. Contains a list of length nSamples, containing for each sample: Counts: list of nPeaks histograms, i.e. for each peak contains vector of integers (read counts); Mids: list of length nPeaks, i.e. for each peak contains histogram mid points (chromosome coordinates); Counts.p: As Counts, but with read counts mapping to the forward strand only; Counts.n: As Counts, but with read counts mapping to reverse strand only; Meta: contains preprocessing meta information. (strand shift, bin length)

Author(s)

Gabriele Schweikert

See Also

dba, summary.DBA, GRanges, scanBam, plotPeak, findOutliers, getNormFactors, compHistDists, detPeakPvals, plotHistDists
Examples

# To build peak profiles you need to store information on your experiment
# in a sample sheet, for example, "Cfp1.csv". You also need bam files,
# which contain the mapped reads for your experiment. The path to the
# bam files should be specified in the csv sample sheet. Lastly you need
# to specify the regions of interest that you want to examine, i.e. the
# peaks.

# Step 1: For this example we provide bam files, peak files and a sample
# sheet in the data package MMDiffBamSubset. We create a new DBA file
# according to the sample sheet "Cfp1.csv":

library('MMDiffBamSubset')
oldwd <- setwd(system.file("extdata", package="MMDiffBamSubset"))
Cfp1 <- dba(sampleSheet="Cfp1.csv",
           minOverlap = 3)

# Step 2: Specify the regions of interest, i.e. the peaks. For this
# example we have run the peak finder Macs [Zhang et al. Genome Biol
# (2008)] on each of the bam file. The xls files containing a subset of
# the peaks are provided and the path to these files is specified in the
# "Cfp1.csv" sample sheet. By calling the function dba, we have also
# generated a list of consensus peaks, which occur in at least
# minOverlap = 3 samples. Let's take the first 1000 consensus peaks:

Peaks <- dba.peakset(Cfp1, bRetrieve=TRUE)
Peaks <- Peaks[1:1000]

# Step 3: Now, we create peak profiles by reading in the reads that map to the
# regions specified in Peaks
Cfp1Profiles <- getPeakProfiles(Cfp1, Peaks, save.files=FALSE)

# To view individual peak profiles use the function plotPeak, e.g.:
plotPeak(Cfp1Profiles, Peak.id=20, NormMethod=NULL)

# Alternative to step 2 you could also get read profiles for any other
# set of regions, for example for these 200 consecutive 100bp regions on
# chromosome 1:
Peaks2 <- GRanges(seqnames = Rle('chr1'),
                 ranges = IRanges(start=seq(3200000, 3219900, 100), width=100))
Cfp1Profiles2 <- getPeakProfiles(Cfp1, Peaks2, save.files=FALSE,
                                 draw.on=FALSE)

setwd(oldwd)
plotHistDists

Description

the generated plots are similar to MA plots, except that the computed distances (MMD) are shown
on the y-axes instead of log fold change.

Usage

plotHistDists(DBA, method = "MMD", group1, group2,
field4X = "NormTotalCounts", bUsePval = FALSE,
 pname = "combined", thresh = 0.05,
 save2file = FALSE, fn.pics, ftype = pdf,
xlim = NULL, ylim = NULL)

Arguments

DBA
method
specify method used to determine distances between histograms (could be MMD,
GMD or Pearson).
group1
group2
field4X
name of list element in DBA$MD that is used for pooling of peaks. (e.g. Norm-
TotalCounts or RawTotalCounts)
bUsePval
logical indicating whether to use FDR (FALSE) or p-value (TRUE) for thresh-
olding.
pnames
name of comparison used to generated p-values, eg. name1 vs name2
thresh
threshold to show significant peaks. (e.g. < 0.05)
save2file
if TRUE plot is saved to pdf file
fn.pics
file name, where the plot should be saved to.
ftype
file format for saving the plot (pdf, postcsript, png)
xlim
the x limits (x1, x2) of the plot. The default value, 'NULL', indicates that the
range of the finite values to be plotted should be used.
ylim
the y limits of the plot.

Author(s)

Gabriele Schweikert

See Also

getPeakProfiles, getNormFactors, compHistDists, detPeakPvals
Examples

# load DBA objects with peak profiles and pairwise distances
data(Cfp1Dists)

# determine empirical p-values:
group1 <- c("WT.AB2", "Resc.AB2")
group2 <- c("Null.AB2")
Cfp1Pvals <- detPeakPvals(Cfp1Dists, group1=group1, group2=group2,
    name1='Wt/Resc', name2='Null')

# plot distances and peaks which are significantly different:
plotHistDists(Cfp1Pvals, group1=group1, group2=group2)

plotPeak 

plot read enrichment profiles at a specific peak for all specified samples.

Description

plot read enrichment profiles at a specific peak for all specified samples.

Usage

plotPeak(DBA, Peak.id, Sample.ids = NULL, NormMethod = 'DESeq',
    plot.input = TRUE, fieldname = "PeakRawHists",
    save2file = FALSE, fn.pics)

Arguments

DBA 
    DBA object, after running getPeakProfiles Specifically, it uses the element MD,
    which should contain an element called according to fieldname.
Peak.id 
    integer specifying the index of the peak to be drawn.
Sample.ids 
    sample ids (as in Cfp1$samples$SampleID)
NormMethod 
    specify which normalization method should be used, currently only the 'DESeq'
    method [3] is implemented. Note, that unless NormMethod=NULL, getNorm-
    Factors has to be called first.
plot.input 
    TRUE, if the input (control) should be included on the plot
fieldname 
    name of list element in DBA$MD that is used for plotting of peak. (e.g. PeakRawHists)
save2file 
    if TRUE plot is saved to pdf file
fn.pics 
    name of pdf file, to which the plot will be saved to.

Author(s)

Gabriele Schweikert
plotPeak

See Also

getPeakProfiles, getNormFactors, plotHistDists

Examples

# load DBA objects with peak profiles
data(Cfp1Profiles)
plotPeak(Cfp1Profiles, Peak.id=20, NormMethod=NULL)

# plot normalized profiles of WT.AB2 and Resc.AB2 samples, don't plot
# the input:
Cfp1Norm <- getNormFactors(Cfp1Profiles)
Sample.ids <- c("WT.AB2", "Resc.AB2")
plotPeak(Cfp1Norm, Peak.id=20, Sample.ids=Sample.ids,
        NormMethod='DESeq', plot.input = FALSE)
Index

*Topic datasets
  Cfp1Dists, 3
  Cfp1Profiles, 4

*Topic package
  MMDiff-package, 2

boxplot, 10
Cfp1Dists, 3
Cfp1Profiles, 4
comphistdists, 5, 9, 13, 15

dba, 4, 5, 13
detPeakPvals, 7, 8, 13, 15

findOutliers, 7, 9, 11, 13
getNormFactors, 7, 9, 10, 13, 15, 17
getPeakProfiles, 7, 9–11, 12, 15, 17
GRanges, 13

MMDiff (MMDiff-package), 2
MMDiff-package, 2

plotHistDists, 7, 9, 13, 15, 17
plotPeak, 7, 9, 11, 13, 16

scanBam, 13
summary.DBA, 4, 5, 13