Package 'methylaction'

May 25, 2015

Maintainer Jeffrey Bhasin <jeffb@case.edu>

Author Jeffrey Bhasin <jeffb@case.edu>

Version 2.0

License MIT

Title MethylAction: Differentially Methylated Regions (DMRs) from MBD-isolated Genome Sequencing (MiGS/MBD-seq)

Description Performs a two-stage testing approach on non-overlapping windows genome-wide to detect differentially methylated regions (DMRs). The statistical DMRs called by differences in group means are then divided into ``frequent'' and ``other'' DMRs to aid in interpretation. Multi-group statistical testing is handled with an analysis of deviance (ANODEV) approach using the GLM functionality of DESeq. Provides functions for bootstrapping the entire method to establish false discovery rates (FDRs).

Depends data.table,
GenomicRanges,
IRanges,
DESeq,
goldmine

Imports parallel,
RColorBrewer,
DESeq,
GenomicAlignments,
Repitools,
reshape,
Rsubread,
R.utils,
ggbio

R topics documented:

methylaction-package 2
getCounts 2
getReads 3
maBed 3
maHeatmap 4
maKaryogram 4
maPerm 5

maPermFdr 6
maPermMerge 6
maSummary 7
maTracks 7
methylaction 8
readSampleInfo 9

Index **10**

`methylaction-package` *MethylAction: Differentially Methylated Regions (DMRs) from MBD-isolated Genome Sequencing (MiGS/MBD-seq)*

Description

MethylAction: Differentially Methylated Regions (DMRs) from MBD-isolated Genome Sequencing (MiGS/MBD-seq)

`getCounts` *Produce a GRanges with counts of overlapping reads for a set of ranges*

Description

Compute counts in non-overlapping windows genome-wide on the chromosomes given in chrs and return as a GRanges object. The values() of the GRanges will contain a table of counts for each sample at each window.

Usage

```
getCounts(samp, reads, ranges = NULL, chrs = NULL, winsize = 50,
  ncore = 1)
```

Arguments

`samp`	Sample data.frame from readSampleInfo()
`reads`	Output from getReads()
`ranges`	Count within regions given by a GRanges object only
`chrs`	Character vector of chromosome names to select
`winsize`	Size of the non-overlapping windows
`ncore`	Number of parallel processes to use

Value

A GenomicRanges object with values() containing a table of counts for each sample at each window.

getReads — *Store GenomicRanges of BAM reads in a list*

Description

Store positions of BAM reads in a list of GRanges objects

Usage

```
getReads(samp, chrs, fragsize, ncore)
```

Arguments

samp	Sample data.frame from readSampleInfo()
chrs	Character vector of chromosome names to select
fragsize	Average fragment length from the sequencing experiment. Reads will be extended up to this size when computing coverage. If set to the value "paired", valid mate pairs for a paired-end sequencing experiment will be loaded instead.
ncore	Number of parallel processes to use

Value

A list of GenomicRanges objects where the ranges represent either extended reads or mate pairs from a BAM file.

maBed — *Write BED file of DMR regions*

Description

Creates a BED file suitable for uploading as a custom track to the UCSC genome browser.

Usage

```
maBed(ma, file)
```

Arguments

ma	Output list from a run of methylaction()
file	Name of BED file to create

Value

Writes BED file to disk.

maHeatmap *Heatmap of the differentially methylated regions (DMRs) found by a run of methylaction()*

Description

Will plot a heatmap of the DMRs based on the normalized read counts. The square root of the mean per-window normalized read count is used so DMRs of different lengths are comparable.

Usage

```
maHeatmap(ma, frequentonly = TRUE, bias = 2, file = NULL)
```

Arguments

ma	Output object from methylaction()
frequentonly	Only plot for DMRs where "frequent" is TRUE
bias	Bias setting for the color scale
file	Where to save the image (PNG format), if NULL, will print to current graphics device

Value

Saves plot to disk or outputs to graphics device

maKaryogram *Karyogram of the differentially methylated regions (DMRs) found by a run of methylaction()*

Description

Will plot a karyogram of the DMRs. A black line above shows regions of coverage by the sequencing experiment.

Usage

```
maKaryogram(ma, reads, frequentonly = TRUE, patt = NULL, colors = NULL,
  file = NULL)
```

Arguments

ma	Output object from methylaction()
reads	Preprocessed reads/fragments data from getReads()
frequentonly	Only plot for DMRs where "frequent" is TRUE
patt	Character vector of patterns to restrict plot to
colors	Character vector of custom colors (as hex codes) for each pattern in patt
file	Where to save the image (PDF format), if NULL, will print to current graphics device

Value

Saves plot to disk or outputs to graphics device

`maPerm` *Permute or Bootstrap DMR Detection*

Description

Will perform permutations or bootstraps after methylaction() has been called. See also maPermMerge() and maPermFdr().

Usage

```
maPerm(ma, reads, nperms, combos = NULL, save = T, perm.boot = F,
  ncore = 1)
```

Arguments

`ma`	Output object from methylaction()
`reads`	Preprocessed reads/fragments data from getReads()
`nperms`	Number of permutation (or bootstrap) iterations to perform
`combos`	A matrix of pre-set combinations to use. Useful for smaller sample sizes where there are only a limited number of possible combinations. The row should have ncols = number of samples, and each row represents a re-ordering of samples into groups (where the groups will be set in the order they appear in the "samp" given to methylaction())
`save`	If true, save an RData file of the permutations. These can be merged using maPermMerge(). Useful for running permutations across multiple computers or in a cluster environment. If FALSE, will return the permutation results.
`perm.boot`	If nperms > 0 and if TRUE, perform bootstrapping (sampling with replacement). Otherwise, perform permutations (sampling without permutations)
`ncore`	Number of parallel processes to use

Value

A list of DMRs arising from each permutation or bootstrap. Will save as an RData if save==TRUE.

maPermFdr *Compute false discovery rates (FDRs) from permuted DMR lists*

Description

Using DMRs arising from permutations saved using maPerm(), compute FDRs for each pattern.

Usage

```
maPermFdr(ma, maperm, recut.p = 0.05, wide = FALSE)
```

Arguments

ma	Output object from methylaction()
maperm	Output object from maPerm()
recut.p	ANODEV p-value cutoff to use
wide	Produce a more readable "wide" format summary table, sorted by "frequent" FDR

Value

A table of FDRs for each pattern, stratified by "frequent" status

maPermMerge *Merge permutations generated and saved by maPerm()*

Description

The function maPerm() with save=TRUE will save permutations into RData files. Move all these RData files into the same directory, and indicate this directory using "dir". The output from maPermMerge() will be a list of DMRs detected in all permutations, and can be given to maPermFdr() to compute false discovery rates (FDRs).

Usage

```
maPermMerge(dir = ".")
```

Arguments

dir	Directory containing RData files saved by maPerm() where save was equal to TRUE

Value

A list of DMRs arising from each permutation or bootstrap, merged across all the detected RData files.

maSummary — *Summary stats for a run of methylaction()*

Description

Will return information about number of windows/regions that pass cutoffs at each stage of the analysis. Useful for parameter tuning.

Usage

```
maSummary(ma)
```

Arguments

ma	Output object from methylaction()

Value

A data.frame with the summary statistics

maTracks — *Write BED and BIGWIG files for normalized, filter-passed window count values*

Description

Creates a BED file suitable for uploading as a custom track to the UCSC genome browser.

Usage

```
maTracks(ma, reads, path = ".", bigwig = FALSE, ncore = 1)
```

Arguments

ma	Output list from a run of methylaction()
reads	Preprocessed reads/fragments data from getReads()
path	Folder to save the files in, will create if does not exist
bigwig	If TRUE, convert the BED to BIGWIG files, requires wigToBigWig in $PATH (obtain from the Jim Kent source tree)
ncore	Number of parallel processes to use

Value

Writes BED file to disk and optionally converts to a bigWig file.

`methylaction` *Detect differentially methylated regions (DMRs) from windowed read counts from MBD-isolated genome sequencing (MiGS/MBD-seq) and similar techniques*

Description

After the counts have been pre-processed, this function performs all the analysis. Detailed results from intermediate steps are stored in the output list object to analyze method performance and provide input for the summary, export, and plotting functions.

Usage

```
methylaction(samp, counts, reads, poifdr = 0.1, stageone.p = 0.05,
  anodev.p = 0.05, post.p = 0.05, freq = 2/3, minsize = 150,
  joindist = 200, adjust.var = NULL, nperms = 0, perm.boot = F,
  ncore = 1)
```

Arguments

`samp`	Sample data.frame from readSampleInfo()
`counts`	Preprocessed count data from getCounts()
`reads`	Preprocessed reads/fragments data from getReads()
`poifdr`	False discovery rate to use during initial filtering and frequency calling. Changing this value will change the threshold used for calling the presence of methylation.
`stageone.p`	P-value cutoff for the pairwise testing in stage one.
`anodev.p`	P-value cutoff for the analysis of deviance (ANODEV) in stage two testing.
`post.p`	P-value cutoff for post-tests in stage two testing.
`freq`	Fraction of samples within groups that must agree with respect to methylation status in order for "frequent" to be "TRUE" in the output DMR list.
`minsize`	Minimum size of DMRs to report (in bp)
`joindist`	Extend significant windows into DMRs over non-significant stage one windows between them up to this distance large (in bp)
`adjust.var`	Name of a column present in "samp" that will be used as a covariate adjustment in the stage two ANODEV generalized linear model (GLM)
`nperms`	Optional, perform this number of permutations after calling DMRs. Will create a data.table called "FDR" in the output list. See also maPerm(), maPermMerge(), and maPermFdr() for manual permutation running and FDR calculation.
`perm.boot`	If nperms > 0 and if TRUE, perform bootstrapping (sampling with replacement). Otherwise, perform permutations (sampling without permutations)
`ncore`	Number of parallel processes to use

Value

A list containing detailed results from each stage of the analysis.

`readSampleInfo` *Load a CSV containing required information about each sample*

Description

The CSV file must contain the following columns: "sample" - unique sample IDs, "group" - group IDs, "bam" - path to BAM file containing aligned reads for the sample. Columns with other names will be retained but ignored. Note that in subsequent reporting of pattern strings (where each digit represents a group), digits for each group will be ordered in the order they first appear in this samplesheet.

Usage

```
readSampleInfo(file = NULL, colors = NULL)
```

Arguments

`file`	Path to the CSV samplesheet to open. Must contain the columns described above.
`colors`	Vector of colors (one for each group) in same order as groups appear in the sample file. These will be uniform colors used in the plotting functions for these groups. Give colors as hex codes. If none are provided they will be auto-selected using RColorBrewer. Alternatively, if there is a column named "color" in the CSV, then these will always be used.

Value

A data.frame of the samplesheet that will be valid input to the "samp" arguments of other functions.

Index

getCounts, 2
getReads, 3

maBed, 3
maHeatmap, 4
maKaryogram, 4
maPerm, 5
maPermFdr, 6
maPermMerge, 6
maSummary, 7
maTracks, 7
methylaction, 8
methylaction-package, 2

readSampleInfo, 9