Package ‘methylaction’

April 28, 2015

Maintainer Jeffrey Bhasin <jeffb@case.edu>

Author Jeffrey Bhasin <jeffb@case.edu>

Version 2.0

License GPL-3

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

R topics documented:

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

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

Usage

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

Arguments

`samp`	Sample data.frame from readSampleInfo()
`bsgenome`	
`ranges`	
`winsize`	Size of the non-overlapping windows.
`fragsize`	Average fragment length from the sequencing experiment. Reads will be extended up to this size when computing coverage.

Value

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

getCountsFc *Get counts using featureCounts*

Description

desc

Usage

```
getCountsFc(samp, bsgenome = NULL, ranges = NULL, chrs = NULL, fragsize,
  winsize = 50, ncore = 1)
```

Arguments

samp	Sample data.frame from readSampleInfo()
bsgenome	
chrs	
fragsize	Average fragment length from the sequencing experiment. Reads will be extended up to this size when computing coverage.

Value

desc

getReads *Store GenomicRanges of BAM reads in a list*

Description

Will return a list of GenomicRanges objects

Usage

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

Arguments

samp	Sample data.frame from readSampleInfo()
bsgenome	
chrs	
fragsize	Average fragment length from the sequencing experiment. Reads will be extended up to this size when computing coverage.

Value

A list of GenomicRanges objects.

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, dmr.only = F)
```

Arguments

`ma`	Output list from a run of methylaction()
`file`	Name of BED file to create
`dmr.only`	Don’t report regions without a significant pattern (show up listed as NS, default: FALSE)

Value

Writes BED file to disk.

maClustering — *Clustering of samples based on read counts*

Description

Will plot clustering based on some subset of read counts.

Usage

```
maClustering(samp, ma, mincv, type, pdf)
```

Arguments

`samp`	Description of samples from readSampleInfo()
`ma`	Output object from methylaction()
`mincv`	Minimum cv (cv=mean/sd) for a window to be included in the clustering.
`type`	One of "mds" or "hca"
`pdf`	PDF file to output.

Value

Saves plot to disk.

`maCompare` *Compare between various runs of methylaction()*

Description

Reports shared and non-shared regions detected by different runs of methylaction(). Useful for comparing between studies/cohorts and paramater tuning.

Usage

```
maCompare(malist)
```

Arguments

`malist`	List of output objects from methylaction(). Set the names() attribute of this list to unique descriptive names for each run.

Value

A list containing both site by site comparisions and a data.frame summary table of total and shared sites for each pairwise comparision between runs.

`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.

Usage

```
maHeatmap(samp, ma, pdf)
```

Arguments

`samp`	Description of samples from readSampleInfo()
`ma`	Output object from methylaction()
`pdf`	PDF file to output.

Value

Saves plot to disk.

`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 paramater tuning.

Usage

```
maSummary(ma)
```

Arguments

`ma`	Output object from methylaction()

Value

A data.frame with the summary statistics.

`maTable` *Table of differentially methylated regions (DMRs) by pattern*

Description

Will return a table of which patterns were detected and the number of DMRs in each.

Usage

```
maTable(samp, ma)
```

Arguments

`samp`	Description of samples from readSampleInfo()
`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, path = ".", bigwig = FALSE, chrs = NULL, bsgenome = NULL,
  ncore = NULL)
```

Arguments

ma	Output list from a run of methylaction()
path	Folder to save the files in (defulat: current working directory)
bigwig	Convert to BIGWIG files, requires wigToBigWig in $PATH (default: FALSE)
chrs	
bsgenome	
ncore	

Value

Writes BED file to disk.

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

Description

Once 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 and plotting functions.

Usage

```
methylaction(samp, counts, winsize, poifdr, stageone.p, joindist, anodev.p,
  post.p, minsize = 150, nperms = 0, ncore = 1)
```

Arguments

`samp`	Description of samples from readSampleInfo()
`counts`	Preprocessed count data from getCounts()
`winsize`	Size of the windows used when counting.
`poifdr`	False discovery rate to use during initital filtering.
`stageone.p`	P-value cutoff for stage one testing.
`anodev.p`	P-value cutoff for the analysis of deviance (ANODEV) in stage two testing (ignored for two group comparisions).
`post.p`	P-value cutoff for post-tests (or for the single test stage two test in the two group case).
`minsize`	Minimum size for a reported region.
`ncore`	Number of cores to use.
`bsgenome`	b-string genome (bsgenome) object for the genome
`fragsize`	The average fragment length selected for in the sequencing experiment (used to extend reads when re-counting for regions in stage two testing).

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 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 provided they will be auto-selected with RColorBrewer. If there is a column named "color" in the CSV, then this will always be used.

Value

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