Package 'methylscaper'

May 30, 2024

Type Package

Title Visualization of Methylation Data

Description methylscaper is an R package for processing and visualizing data jointly profiling methylation and chromatin accessibility (MAPit, NOMe-seq, scNMT-seq, nanoNOMe, etc.). The package supports both single-cell and single-molecule data, and a common interface for jointly visualizing both data types through the generation of ordered representational methylation-state matrices. The Shiny app allows for an interactive seriation process of refinement and re-weighting that optimally orders the cells or DNA molecules to discover methylation patterns and nucleosome positioning.

Version 1.12.0

Depends R (>= 4.1.0)

License GPL-2

Encoding UTF-8

Imports shiny, shinyjs, seriation, BiocParallel, seqinr, Biostrings, pwalign, Rfast, grDevices, graphics, stats, utils, tools, methods, shinyFiles, data.table, SummarizedExperiment

RoxygenNote 7.3.1

Suggests BiocStyle, knitr, rmarkdown, devtools, R.utils

VignetteBuilder knitr

biocViews DNAMethylation, Epigenetics, Sequencing, Visualization, SingleCell, NucleosomePositioning

Maintainer Rhonda Bacher <rbacher@ufl.edu>

git_url https://git.bioconductor.org/packages/methylscaper

git_branch RELEASE_3_19

git_last_commit 27c9848

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-29

Author Bacher Rhonda [aut, cre], Parker Knight [aut]

Contents

`check_package` *Check if a package is installed*

Description

Check if a package is installed

Usage

```
check_package(package)
```

Arguments

`package`	Name of the package.

Value

Message is returned if package not installed.

forceReverse *Force reversal of a subset of the ordering*

Description

This reverses a subset of the ordering, as determined by the user. By default, the entire ordering is reversed.

Usage

```
forceReverse(
  orderObject,
  reverseStart = 1,
  reverseEnd = length(orderObject$order1)
)
```

Arguments

orderObject	An object of class orderObject, generated with the initialOrder function.
reverseStart	The first index to be included in the reversal.
reverseEnd	The last index to be included in the reversal.

Value

The new complete ordering, with the reversal applied.

Examples

```
data(singlemolecule_example)

orderObj <- initialOrder(singlemolecule_example, Method = "PCA")
# reorder first 50 cells/molecules (rows)
orderObj$order1 <- refineFunction(orderObj, 1, 50)
orderObj$order1 <- forceReverse(orderObj, 1, 50)
```

human_bm *Human gene symbols and positions*

Description

library(biomaRt) #v2.44.4 ensembl <- useMart("ensembl") # GRCh38 ensembl <- useDataset("hsapiens_gene_ensembl",mar
my_chr <- c(1:22, 'M', 'X', 'Y') human_bm <- getBM(attributes=c('chromosome_name', 'start_position',
'end_position', 'hgnc_symbol'), filters = 'chromosome_name', values = my_chr, mart=ensembl)

Usage

```
data(human_bm)
```

Format

An object of class `data.frame` with 60580 rows and 4 columns.

---

initialOrder *Ordering the molecules/reads*

Description

This function performs the weighted seriation procedure described in the methylscaper manuscript if the method is set to "PCA". The data may also be ordered using a given seriation method from the seriation R package. The weighting is done between a designated start and end base pair chosen by the user, and the weight can be done on the endogenous methylation or the accessibility.

Usage

```
initialOrder(
  dataIn,
  Method = "PCA",
  weightStart = NULL,
  weightEnd = NULL,
  weightFeature = "red",
  updateProgress = NULL
)
```

Arguments

`dataIn`	A list object containing two elements labelled gch and hcg (already pre-processed.)
`Method`	Indicates the seriation method to use. The default option is "PCA", which orders the data using a weighted first principal component approach. Any seriation method provided in the `seriation` package is also valid input.
`weightStart`	Index of the first column used in the weighted seriation.
`weightEnd`	Index of the last column used in the weighted seriation.
`weightFeature`	Indicates whether to weight the GCH or HCG data. Valid input to weight the GCH is 'gch', 'acc', or 'yellow'. To weight the HCG, valid input for this option is 'hcg', 'met', or 'red'.
`updateProgress`	A function to handle the progress bar for the Shiny app. Should not be used when using the function independently.

Value

An object of class `orderObject`, which contains the generated ordering ($order1) and a clean data matrix ($toClust) to be passed into the plotting function plotSequence().

Examples

```
data(singlemolecule_example)

orderObj <- initialOrder(singlemolecule_example)
```

`methylscaper` *methylscaper*

Description

Runs the methylscaper Shiny app.

Usage

```
methylscaper()
```

Value

This starts up the shiny app interface for methylscaper.

Examples

```
# methylscaper()
```

`methyl_average_status` *Calculate the average methylation/accessibility status across all cells/molecules.*

Description

Calculate the average methylation/accessibility status across all cells/molecules.

Usage

```
methyl_average_status(orderObject, window_length = 20, makePlot = TRUE, ...)
```

Arguments

`orderObject`	An object of class `orderObject`
`window_length`	Length of the window to be used to compute a moving average. Default is 20.
`makePlot`	Logical, indicates whether to generate a line plot of average status.
`...`	Addition parameters used by the `plot` function.

Value

The proportion of methylated bases for each cell/molecule within a defined moving window. Output is a list with elements "meth_avg" and "acc_avg", indicating endogenous or accessible methylation respectively.

Examples

```
data(singlemolecule_example)

orderObj <- initialOrder(singlemolecule_example, Method = "PCA")
methyl_average_status(orderObj, makePlot = TRUE)
```

`methyl_percent_sites` *Calculates the percentage of methylated cells/molecules per site*

Description

Calculates the percentage of methylated cells/molecules per site

Usage

```
methyl_percent_sites(orderObject, makePlot = TRUE, ...)
```

Arguments

`orderObject`	An object of class `orderObject`
`makePlot`	Logical, indicates whether to generate the percentage plot
`...`	Additional parameters used by the `plot` function.

Value

The percent of molecules or cells methylated (endogenous (yellow) or accessible) at each site. Output is a list with names "red" and "yellow". Red represents the endogenous methylation and yellow represents the accessibility. Within each list object is a vector of the percent of cells/molecules methylated. The location of the site is also represent in the form CXX, where XX is the position of the site within the defined region.

Examples

```
data(singlemolecule_example)

orderObj <- initialOrder(singlemolecule_example, Method = "PCA")
methyl_percent_sites(orderObj, makePlot = TRUE)
```

methyl_proportion *Calculate the proportion of methylated bases for each cell/molecule*

Description

Calculate the proportion of methylated bases for each cell/molecule

Usage

```
methyl_proportion(orderObject, type = "yellow", makePlot = TRUE, ...)
```

Arguments

orderObject	An object of class orderObject
type	Indicates which data set to compute proportions for. This should be 'met' or 'hcg' or 'red' for endogenous methylation; 'acc' or 'gch' or 'yellow' for accessibility.
makePlot	Indicates whether to plot a histogram of the proportions across all cells/molcules.
...	Additional parameters used by the hist function.

Value

The proportion of methylated (endogenous (yellow) or accessible) bases for each cell/molecule. Output is vector with length the numbner of cells/molecules and contains a proportion.

Examples

```
data(singlemolecule_example)

orderObj <- initialOrder(singlemolecule_example, Method = "PCA")
methyl_proportion(orderObj, makePlot = TRUE)
```

mouse_bm *Mouse gene symbols and positions*

Description

library(biomaRt) #v2.44.4 ensembl <- useMart("ensembl") # GRCm39 ensembl <- useDataset("mmusculus_gene_ensembl",
my_chr <- c(1:19, 'M', 'X', 'Y') mouse_bm <- getBM(attributes=c('chromosome_name', 'start_position',
'end_position', 'mgi_symbol'), filters = 'chromosome_name', values = my_chr, mart=ensembl)

Usage

```
data(mouse_bm)
```

Format

An object of class `data.frame` with 55365 rows and 4 columns.

`plotSequence` *Generate Sequence Plot*

Description

Generates an ordered sequence plot of methylation data.

Usage

```
plotSequence(
  orderObject,
  plotFast = TRUE,
  blankWidth = NULL,
  Title = "",
  drawLine = TRUE,
  drawKey = TRUE
)
```

Arguments

`orderObject`	An object of class `orderObject` that contains the processed data and the ordering.
`plotFast`	Logical, setting to FALSE will generate a higher quality plot. TRUE generates a lower resolution file, useful to improve speed while testing. For publication quality use plotFast=TRUE.
`blankWidth`	Indicates the amount of space to leave between the two plots
`Title`	The title of the plot.
`drawLine`	Logical, indicates whether to draw a line above the CG/GC sites.
`drawKey`	Logical, indicates whether to draw a key representing a 147bp nucleosome at the bottom of the plot.

Value

Output is two side-by-side heatmaps with the endogenous methylation (HCG) on the left and the acciessibility methylation (GCH) on the right. The tick marks at the top indicate either HCG or GCH sites when drawLine=TRUE. If drawKey=TRUE then a black rectangle key is plot at the bottom of the heatmap that is 147 basepairs long. In the HCG plot, red patches represent methylation between two sites; black patches represent unmethylated bases between two unmethylated sites; and gray patches are base pairs which have one methylated site and one unmethylated site flanking. In the HCG plot, yellow patches represent accessibility between two sites; black patches represent occupied bases between two occupied sites; and gray patches are base pairs which have one methylated site and one unmethylated site flanking.

Examples

```
data(singlemolecule_example)

orderObj <- initialOrder(singlemolecule_example, Method = "PCA")
plotSequence(orderObj)
```

---

prepSC *Process single-cell data*

Description

This function subsets the data and prepares it for visualizing by generating representation methylation-state matrices from single-cell methylation data (for example, sc-MNT data). We assume the data has already been preprocess using the subsetSC function in methylscaper. See the vignette for a more thorough explanation of each parameter. The output should be passed directly to the plotting function.

Usage

```
prepSC(dataIn, startPos = NULL, endPos = NULL, updateProgress = NULL)
```

Arguments

dataIn	A list object containing two elements labelled gch and hcg (already pre-processed.)
startPos	The index of the first position to include in the visualization. If using this within the R console it is recomended to specify the start and end directly. In the Shiny app, a slider will let the user refine these positions.
endPos	The index of the final position to include in the visualization.
updateProgress	A function for generating progress bars in the Shiny app. Should be left NULL otherwise.

Value

The output is a list containing the elements 'gch' and 'hcg. Each is a dataframe with reads/cells on the rows and each column is a base-pair. The matrix is coded as follows: -2: unmethylated GCH or HCG site -1: base pairs between two unmethylated GCH or HCG sites 0: base pairs between mismatching methylation states of two GCH or HCG sites 1: base pairs between two methylated GCH or HCG sites 2: methylated GCH or HCG site

Examples

```
data(singlecell_subset)
prepsc.out <- prepSC(singlecell_subset,
                startPos = 105636488, endPos = 105636993)
```

reads_sm *Example reads from single-molecule experiment*

Description

This dataset was loaded into R using seqinr::read.fasta

Usage

```
data(reads_sm)
```

Format

An object of class `list` of length 293.

refineFunction *Refinement*

Description

Reorders a subset of the methylation data.

Usage

```
refineFunction(orderObject, refineStart, refineEnd, Method = "PCA")
```

Arguments

`orderObject`	An object of class `orderObject`, generated with the `initialOrder` function.
`refineStart`	The index of the first sample (row) used in the refinement.
`refineEnd`	The index of the last sample (row) used in the refinement.
`Method`	The seriation method used to perform the refinement.

Value

The refinement reorders the cells/molecules (rows) between the indicated start and end positions. The function returns the new complete ordering with the refinement applied.

Examples

```
data(singlemolecule_example)

orderObj <- initialOrder(singlemolecule_example, Method = "PCA")
# reordering the first 50 cells/molecules (rows)
orderObj$order1 <- refineFunction(orderObj, 1, 50)
```

`reformatSCE` *This is an internal function for now and a place-holder in case SingleCellExperiment may be used in the future. We may need to update this later. Assumes rownames are formatted like chr_pos and there are two assays. The assay names are 'methylation_met' for endogenous methylation and 'methylation_acc' for accessibility.*

Description

This is an internal function for now and a place-holder in case SingleCellExperiment may be used in the future. We may need to update this later. Assumes rownames are formatted like chr_pos and there are two assays. The assay names are 'methylation_met' for endogenous methylation and 'methylation_acc' for accessibility.

Usage

```
reformatSCE(dataIn)
```

Arguments

`dataIn`	Input SCE object passed to the prepSC() function.

Value

List object containing the met and acc tables.

`ref_seq` *Example reference sequence to align reads to from a single-molecule experiment*

Description

This dataset was loaded into R using seqinr::read.fasta

Usage

```
data(ref_seq)
```

Format

An object of class `list` of length 1.

runAlign *Align the single-molecule data*

Description

Runs the preprocessing methods for single-molecule data.

Usage

```
runAlign(
  ref,
  fasta,
  fasta_subset = seq(1, length(fasta)),
  multicoreParam = NULL,
  updateProgress = NULL,
  log_file = NULL
)
```

Arguments

ref	A reference sequence to align the reads to.
fasta	A list of reads/sequences from a single-molecule experiment (e.g. MAPit)
fasta_subset	(optional) A vector of indices indicating which sequences to process if a subset should be used. Leave this blank if all sequences should be processed.
multicoreParam	(optional) A MulticoreParam object, used to align sequences in parallel.
updateProgress	(optional) Used to add a progress bar to the Shiny app. Should not be used otherwise.
log_file	(optional) String indicating where to save a log of the alignment process. If left NULL, no log is saved. We highly recommend saving a log file.

Value

The output is a list containing the the matrices 'gch' and 'hcg. Each is a dataframe with reads/cells on the rows and each column is a base-pair. The matrix represents the methylation state for cell across all base pairs. The coding is as follows: -2: unmethylated GCH or HCG site -1: base pairs between two unmethylated GCH or HCG sites 0: base pairs between mismatching methylation states of two GCH or HCG sites 1: base pairs between two methylated GCH or HCG sites 2: methylated GCH or HCG site

Examples

```
data(reads_sm)
data(ref_seq)
example_alignedseq <- runAlign(fasta=reads_sm, ref = ref_seq[[1]], fasta_subset = 1:150)
```

singlecell_subset *Example preprocessed single-cell experiment subset*

Description

This data is from GSE109262, and has been pre-processed by methylscaper It contains a small subset of chromosome 19 region from 8947041bp - 8987041bp. The RDS in ext data was made specifically with the two commands: singlecell_subset <- subsetSC("~/Downloads/GSE109262_RAW/", chromosome="19", startPos = 8967041-20000, endPos = 8967041+20000, updateProgress = NULL) saveRDS(singlecell_subset, file="methylscaper/inst/ext/singlecell_subset.rds", compress = 'xz') A version is also saved as RData used running examples in the man pages. save(singlecell_subset, file="methylscaper/data/singlecell_subset.RData", compress = 'xz')

Usage

```
data(singlecell_subset)
```

Format

An object of class `list` of length 2.

singlemolecule_example *Example preprocessed single-molecule experiment*

Description

The RDS in ext data was made specifically with the command: singlemolecule_example <- methylscaper::runAlign(fasta=reads_sm, ref=ref_seq) saveRDS(singlemolecule_example, file="methylscaper/inst/ext/singlemolecule_ compress = 'xz') A version is also saved as RData used running examples in the man pages. save(singlemolecule_example, file="methylscaper/data/singlemolecule_example.RData", compress = 'xz')

Usage

```
data(singlemolecule_example)
```

Format

An object of class `list` of length 2.

subsetSC *Load in methylation data*

Description

This function loads the single-cell files. It takes a path to the data files and a chromosome number as arguments and returns the desired subset of the data. Processing by chromosome is important for speed and memory efficiency. The input files should be tab separated with three columns. The first column is the chromosome, the second is the position (basepair), and the third is the methylation indicator/rate. The folder should contain two subfolders titled met and acc, with the endogenous methylation and accessibility methylation files, respectively.

Usage

```
subsetSC(
  path,
  chromosome,
  startPos = NULL,
  endPos = NULL,
  updateProgress = NULL
)
```

Arguments

path	Path to the folder containing the single-cell files.
chromosome	The chromosome to subset the files to.
startPos	The index of the first position to include in the subsetting. This is optional as further narrowing of the position can be done in the visualization step/tab. In the Shiny app, a slider will let the user refine the positions.
endPos	The index of the final position to include in subset.
updateProgress	A function for generating progress bars in the Shiny app. Should be left NULL otherwise.

Value

The output is RDS files that can be loaded into the visualization tab on the Shiny app

Examples

```
# example not run since needs directory input from user
#subsc.out <- subsetSC("filepath", chromosome=19)
```

Index