Package 'methInheritSim'

February 28, 2026

Type Package

Title Simulating Whole-Genome Inherited Bisulphite Sequencing Data

Description Simulate a multigeneration methylation case versus control
experiment with inheritance relation using a real control
dataset.

Version 1.32.0

Date 2025-07-22

Author Pascal Belleau, Astrid Deschênes and Arnaud Droit

Author@R c(person(``Pascal'', ``Belleau'',
email=``pascal_belleau@hotmail.com'', role=c(``cre'', ``aut'')),
person(``Astrid'', ``Deschênes'', email=``adeschen@hotmail.com'',
role=c(``aut'')), person(``Arnaud'', ``Droit'',
email=``arnaud.droit@crchuq.ulaval.ca'', role=c(``aut'')))

Depends R (>= 3.4)

Imports methylKit, GenomicRanges, Seqinfo, parallel, BiocGenerics,
S4Vectors, methods, stats, IRanges, msm

Suggests BiocStyle, knitr, rmarkdown, RUnit, methylInheritance

Encoding UTF-8

License Artistic-2.0

URL https://github.com/belleau/methInheritSim

BugReports https://github.com/belleau/methInheritSim/issues

VignetteBuilder knitr

biocViews BiologicalQuestion, Epigenetics, DNAMethylation,
DifferentialMethylation, MethylSeq, Software, ImmunoOncology,
StatisticalMethod, WholeGenome, Sequencing

Maintainer Pascal Belleau <pascal_belleau@hotmail.com>

RoxygenNote 6.0.1

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

git_branch RELEASE_3_22

git_last_commit b52e28a

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-02-27

Contents

---

methInheritSim-package

methInheritSim: Simulating Whole-Genome Inherited Bisulphite Sequencing Data

---

Description

This package generates simulations of multigeneration of bisulfite data using a real control dataset.

Author(s)

Pascal Belleau, Astrid Deschênes and Arnaud Droit

Maintainer: Pascal Belleau <pascal_belleau@hotmail.com>

See Also

- [runSim](#) for simulating a multigeneration methylation experiment with inheritance

---

calculateNbDiffCase   *Calculate the number of differentially methylated cases.*

Description

Identify the number of differentially methylated cases.

Usage

```
calculateNbDiffCase(nbCase, propDiff, propDiffSd)
```

Arguments

nbCase   a positive `integer`, the number of cases.

propDiff   a double superior to `0` and inferior or equal to 1, the mean value for the propor-
                  tion of samples that will have, for a specific position, differentially methylated
                  values. It can be interpreted as the penetrance.

propDiffSd   a non-negative `double`, the standard deviation associated to the `propDiff`

Value

a `integer`, the number of differentially methylated cases.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Fix seed to have reproducible results
set.seed(3122)

## Obtained the number of differential cases
methInheritSim:::calculateNbDiffCase(nbCase = 8,
    propDiff = 0.8, propDiffSd = 0.2)
```

---

createSampleID   *Generate the samples ID for the simulated dataset.*

Description

Generate the samples ID for the simulated dataset. The standard format of the samples ID is :
"F[Number for the generation]_[Number for the sample] _[OC for case or C for control]"

Usage

```
createSampleID(nbGeneration, nbSample)
```

Arguments

nbGeneration	a positive integer, the number of generations simulated.
nbSample	a positive integer, the number of controls (CTRL) and cases in the simulated dataset.

Value

a list containing a list of sample ID for each generation.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Create sample ID
methInheritSim:::createSampleID(nbGeneration = 3, nbSample = 6)
```

---

dataSimExample	*A* list *containing methylation information used by some internal functions (for demo purpose.*

Description

A list containing methylation information used by some internal functions (for demo purpose.

Usage

```
data(dataSimExample)
```

Format

a list containing:

- stateInfo a GRanges, a synthetic chromosome as generated by getSyntheticChr function.
- stateDiff a list containing:
  - stateDiff a vector of integer (0 and 1) with length corresponding the length of stateInfo. The vector indicates, using 1, the positions where the CpG sites are differentially methylated.
  - stateInherite a vector of integer (0 and 1) with length corresponding the length of stateInfo. The vector indicates, using 1, the positions where the CpG values are inherited.
- treatment a vector of integer (0 and 1) with length corresponding the number of samples. The vector indicates which samples are control (0) which samples are case (1).
- sample.id a list of 3 list. Each entry of the list correspond to one generation (first entry = first generation, etc..). Each list contains a list of 12 entries each containing a string of character, the name of the sample.

Value

a `list` containing:

- `stateInfo` a `GRanges`, a synthetic chromosome as generated by `getSyntheticChr` function.
- `stateDiff` a `list` containing:
    - `stateDiff` a vector of `integer` (0 and 1) with length corresponding the length of `stateInfo`. The vector indicates, using 1, the positions where the CpG sites are differentially methylated.
    - `stateInherite` a vector of `integer` (0 and 1) with length corresponding the length of `stateInfo`. The `vector` indicates, using 1, the positions where the CpG values are inherited.
- `treatment` a vector of `integer` (0 and 1) with length corresponding the number of samples. The `vector` indicates which samples are control (0) which samples are case (1).
- `sample.id` a `list` of 3 `list`. Each entry of the `list` correspond to one generation (first entry = first generation, etc..). Each `list` contains a `list` of 12 entries each containing a string of `character`, the name of the sample.

See Also

- [runSim](#) for running a simulation analysis using methylKit info as input

Examples

```
## Loading dataset
data(dataSimExample)

## Identify differentially methylated sites and among those, the ones
## that are inherited
methInheritSim:::getDiffMeth(stateInfo = dataSimExample$stateInfo,
    rateDiff = 0.2, minRate = 0.3,propInherite = 0.3)
```

---

estBetaAlpha	*Estimate the alpha parameter of a Beta distribution*

Description

Estimate the alpha parameter from the mean and the variance of a Beta distribution.

Usage

```
estBetaAlpha(meanCtrl, varCtrl, minVal = 1e-06)
```

Arguments

meanCtrl	a double, the mean of the controls (CTRL) at a specific CpG site.
varCtrl	a double, the variance of the controls (CTRL) at a specific CpG site.
minVal	a double, the minimum value accepted for the mean value. If `meanCtrl` is smaller than `minVal`, then `minVal` is used in the calculation of the alpha parameter. Default: `1e-06`.

Value

a `double`, the alpha parameter of a Beta distribution.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Estimate alpha parameters with mean = 0.5 and variance = 0.1
methInheritSim:::estBetaAlpha(meanCtrl = 0.5, varCtrl = 0.1)
```

---

estBetaBeta   *Estimate the beta parameter of a beta distribution*

Description

Estimate the beta parameter from the mean and the variance of a beta distribution.

Usage

```
estBetaBeta(meanCtrl, varCtrl, minVal = 1e-06)
```

Arguments

meanCtrl	a `double`, the mean of the controls (CTRL) at a specific CpG site.
varCtrl	a `double`, the variance of the controls (CTRL) at a specific CpG site.
minVal	a `double`, the minimum value accepted for the mean value. If `meanCtrl` is smaller than `minVal`, then `minVal` is used in the calculation of the beta paramter. Default: `1e-06`.

Value

a `double`, the beta parameter of a Beta distribution.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Estimate beta parameters with mean = 0.5, variance = 0.1
methInheritSim:::estBetaBeta(meanCtrl=0.5, varCtrl=0.1)
```

---

fixSeed   *Fix seed value.*

Description

Fix seed value when specified value is -1.

Usage

```
fixSeed(vSeed)
```

Arguments

vSeed   a integer, a seed used when reproducible results are needed. When a value
                 inferior or equal to zero is given, a random integer is used.

Value

a double, the seed value.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Return vSeed value when value is not -1
methInheritSim:::fixSeed(vSeed = 10)

## Return new value when value is -1
methInheritSim:::fixSeed(vSeed = -1)
```

---

getDiffCase   *Get the C/T proportion at a selected site, differentially methylated or
                            not, for all cases*

Description

Simulate the proportion of C/T for each case at a selected site, differentially methylated or not.

Usage

```
getDiffCase(ctrlMean, ctrlVar, selectedAsDM, nbCase, sDiff, nbDiffCase)
```

Arguments

`ctrlMean`	a double, the mean of the CTRL at the site.
`ctrlVar`	a double, the variance of the CTRL at the site.
`selectedAsDM`	a integer, 1 if the site is selected as differentially methylated, otherwise 0.
`nbCase`	a integer, the number of cases.
`sDiff`	a non-negative double included in [0,1], the proportion of C/T for a case differentially methylated that follows a beta distribution where the mean is shifted of `vDiff` from the CTRL distribution.
`nbDiffCase`	an integer, the number of cases differentially methylated.

Value

a vector containing 3 + nbCase entries of type double:

- The mean proportion of C/T of the differentially methylated cases
- The number of cases simulated using shifted distribution
- The number of cases simulated using the control distribution
- The proportion of C/T for each case

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Fix seed to obtain replicable results
set.seed(2010)

## Get the proportion of C/T for each case at a specific site.
methInheritSim:::getDiffCase(ctrlMean = 0.9814562, ctrlVar =
0.0003607153, selectedAsDM = 0, nbCase=6, sDiff = 0.8,
nbDiffCase = round(6 * 0.9))
```

---

getDiffMeth	*Identify differentially methylated sites and among those, the ones that are inherited.*

Description

Identify the site positions where the cases are differentially methylated and, among those, the one that are inherited.

Usage

```
getDiffMeth(stateInfo, rateDiff, minRate, propInherite, c = 1, b = -0.1,
  endLength = 1000)
```

Arguments

stateInfo	a GRanges that contains the CpG (or methylated sites). The GRanges have four metadata from the real dataset:

- chrOri, the chromosome from the real dataset
- startOri, the position of the site in the real dataset
- meanCTRL, the mean of the control in the real dataset
- varCTRL, the variance of the control in the real dataset

rateDiff	a positive double inferior to 1, the mean of the chance that a site is differentially methylated.
minRate	a non-negative double inferior to 1, the minimum rate for differentially methylated sites.
propInherite	a non-negative double inferior or equal to 1, the proportion of differentially methylated regions that are inherated.
c	a positive double, a factor in the formula to compute the probabylity of site to be diffentally methylated in a differentially methylated region. The probability formula of site in differentially methylated region is c * exp(b * log(distance with the preceding sites)) Default: 1.0.
b	a negative double, a factor in the formula to compute the probabylity of site to be diffentally methylated in a differentially methylated region. The probability formula of site in differentially methylated region is c * exp(b * log(distance with the preceding sites)). Default: -1e-01.
endLength	a positive integer, when the distance with the preceding sites in a differentially methylated region is larger than endLength, the differentially methylated region is finished. Default: 1000.

Value

a list containing the 2 following elements:

- stateDiff a vector of integer (0 and 1) with length corresponding the length of stateInfo. The vector indicates, using 1, the positions where the CpG sites are differentially methylated.
- stateInherite a vector of integer (0 and 1) with length corresponding the length of stateInfo. The vector indicates, using 1, the positions where the CpG values are inherited.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Load dataset containing a list of objects used by
## methInheritSim internal functions
data(dataSimExample)

## Identify differentially methylated sites and among those, the ones
## that are inherited
methInheritSim:::getDiffMeth(stateInfo =
    dataSimExample$stateInfo, rateDiff = 0.3, minRate = 0.3,
    propInherite = 0.3)
```

| getSim | *Simulate the proportion of C/T at each site of synthetic CHR for each control and case* |

Description

For each control and case, generate the proportion of C/T at each of the synthetic CHR.

Usage

```
getSim(nbCtrl, nbCase, generation, stateInfo, stateDiff, stateInherite,
    diffValue, propDiff, propDiffsd, propInheritance, propHetero)
```

Arguments

nbCtrl	a positive `integer`, the number of controls.
nbCase	a positive `integer`, the number of cases.
generation	a positive `integer`, the number of generations.
stateInfo	a `GRanges` object, the synthetic chromosome generated by `getSyntheticChr` function.
stateDiff	a `vector` of `integer` (0 and 1) with length corresponding the length of `stateInfo`. The `vector` indicates, using a 1, the positions where the CpG sites are differentially methylated.
stateInherite	a `vector` of `integer` (0 and 1) with length corresponding the length of `stateInfo`. The `vector` indicates, using a 1, the positions where the CpG values are inherited.
diffValue	a non-negative `double` between between [0,1], the proportion of C/T for a case differentially methylated following a beta distribution where the mean is shifted of `diffValue` from the CTRL distribution.
propDiff	a `double` superior to `0` and inferior or equal to 1, the mean value for the proportion of samples that will have, for a specific position, differentially methylated values. It can be interpreted as the penetrance.
propDiffsd	a non-negative `double`, the standard deviation associated to the `propDiff`.
propInheritance	
	a non-negative `double` between [0,1], the proportion of case that inherit differentially methylated sites.
propHetero	a non-negative `double` between [0,1], the reduction of `vDiff` for the second and following generations.

Value

a `GRangesList`, the object contains information about the simulation. The file have four metadata related to the real dataset:

- meanDiff, the means of the shifted distribution.
- meanCTRL, the means of the control distribution.
- partitionCase, the number of cases simulated using the shifted distribution.
- partitionCtrl, the number of cases simulated using the control distribution and a metadata for each cases and controls the proportion of C/T.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Fix seed to have reproducible results
set.seed(312)

## Load dataset
data("samplesForChrSynthetic")

## Generate a stateInfo object using samples
stateInformation <- methInheritSim:::getSyntheticChr(methInfo =
    samplesForChrSynthetic, nbBlock = 1, nbCpG = 3)

## Generate a stateDiff and stateInherite objects with length corresponding
## to nbBlock * nbCpG from stateInformation
stateDiff  <- c(1, 0, 1)
stateInherite <- c(1, 0, 0)

## Create a simulation using stateInformation, stateDiff and stateInherite
methInheritSim:::getSim(nbCtrl = 3, nbCase = 2, generation = 3,
    stateInfo = stateInformation, stateDiff = stateDiff,
    stateInherite = stateInherite, diffValue = 10,
    propDiff = 0.8, propDiffsd = 0.2, propInheritance = 0.8,
    propHetero = 0.1)
```

---

getSyntheticChr   *Create a synthetic chromosome with the CTRL genome*

Description

Create a synthetic chromosome with the sampling of a specified number of blocks and a specified number of consecutive CpG.

Usage

```
getSyntheticChr(methInfo, nbBlock, nbCpG)
```

Arguments

methInfo   is object of class `methylBase`, the CpG information from controls (CTRL) that will be used to create the sythetic chromosome. The object can also contain information from cases but only the controls will be used.

nbBlock   a positive `integer`, the number of blocks used for sampling.

nbCpG   a `integer`, the number of consecutive CpG positions used for sampling from `methInfo`.

Value

a `GRanges` object, the synthetic chromosome.

Author(s)

Pascal Belleau

Examples

```
## Load methyl information
data(samplesForChrSynthetic)

## Ensure results are reproducible
set.seed(32)

## Create synthetic chromosome
methInheritSim:::getSyntheticChr(methInfo = samplesForChrSynthetic,
nbBlock = 10, nbCpG = 20)
```

---

runOnEachSynCHR   *Simulate a multigeneration methylation experiment with inheritance*
                           on each synthetic chromosome.

Description

Simulate a multigeneration methylation case versus control experiment with inheritance relation
using a real control dataset.

Usage

```
runOnEachSynCHR(methData, nbSynCHR, nbSimulation, nbBlock, nbCpG, nbGeneration,
  vNbSample, vpDiff, vpDiffsd, vDiff, vInheritance, rateDiff, minRate,
  propInherite, propHetero, keepDiff, outputDir, fileID, minReads, maxPercReads,
  meanCov, context, assembly, saveGRanges, saveMethylKit, runAnalysis, nbCores,
  vSeed)
```

Arguments

methData	an object of class `methylBase`, the CpG information from controls (CTRL) that will be used to create the synthetic chromosome. The `methData` object can also contain information from cases but only the controls are used.
nbSynCHR	a positive `integer`, the number of distinct synthetic chromosomes that will be generated.
nbSimulation	a positive `integer`, the number of simulations generated for each parameter (`vNbSample`, `vpDiff`, `vDiff` and `vInheritance`). The total number of simulation is nbSimulation * length(vNbSample) * length(vpDiff) * length(vInheritance)
nbBlock	a positive `integer`, the number of blocks used for sampling.
nbCpG	a positive `integer`, the number of consecutive CpG positions used for sampling from `methInfo`.
nbGeneration	a positive `integer`, the number of generations simulated.
vNbSample	a `vector` of distinct positive `integer`, the number of controls (CTRL) and cases in the simulated dataset. In the simulated dataset, the number of CTRL equals the number of cases. The number of CTRL do not need to be equal to the number of Case in the real `methData` dataset.

vpDiff	a vector of distinct double superior to 0 and inferior or equal to 1, the mean value for the proportion of samples that will have, for a specific position, differentially methylated values. It can be interpreted as the penetrance. Note that vpDiff and vpDiffsd must be the same length.
vpDiffsd	a vector of a non-negative double, the standard deviation associated to the vpDiff. Note that vpDiff and vpDiffsd must be the same length.
vDiff	a vector of distinct non-negative double included in [0,1], the proportion of C/T for a case differentially methylated that follows a beta distribution where the mean is shifted by vDiff from the CTRL distribution.
vInheritance	a vector of distinct non-negative double included in [0,1], the proportion of cases that inherits differentially methylated sites.
rateDiff	a positive double inferior to 1, the mean of the chance that a site is differentially methylated.
minRate	a non-negative double inferior to 1, the minimum rate for differentially methylated sites.
propInherite	a non-negative double inferior or equal to 1, the proportion of differentially methylated regions that are inherated.
propHetero	a non-negative double between [0,1], the reduction of vDiff for the second and following generations.
keepDiff	a logical, when TRUE, the differentially methylated sites will be the same for all simulated datasets. Datasets generated using differents parameter values from vector parameters (vpDiff, vDiff and vInheritance) wil all have the same differentially methylated sites.
outputDir	a string of character or NULL, the path where the files created by the function will be saved. When NULL, the files are saved in a directory called "outputDir" that is located in the current directory.
fileID	a string of character, a identifiant that will be included in each output file name. Each output file name is composed of those elements, separated by "_":

- a type name, ex: methylGR, methylObj, etc..
- a fileID
- the chromosome number, a number between 1 and nbSynCHR
- the number of samples, a number in the vNbSample vector
- the mean proportion of samples that has, for a specific position, differentially methylated values, a number in the vpDiff vector
- the proportion of C/T for a case differentially methylated that follows a shifted beta distribution, a number in the vDiff vector
- the proportion of cases that inherits differentially sites, a number in the vInheritance vector
- the identifiant for the simulation, a number between 1 and nbSimulation
- the file extension ".rds"

minReads	a positive integer, sites and regions having lower coverage than this count are discarded. The parameter corresponds to the lo.count parameter in the methylKit package.
maxPercReads	a double between [0,100], the percentile of read counts that is going to be used as upper cutoff. Sites and regions having higher coverage than maxPercReads are discarded. This parameter is used for both CpG sites and tiles analysis. The parameter correspond to the hi.perc parameter in the methylKit package.

meanCov	a positive `integer`, the mean of the coverage at the simulated CpG sites.
context	a string of `character`, the short description of the methylation context, such as "CpG", "CpH", "CHH", etc..
assembly	a string of `character`, the short description of the genome assembly, such as "mm9", "hg18", etc..
saveGRanges	a `logical`, when true, the package save two files type. The first generate for each simulation contains a `list`. The length of the `list` corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a GRangesList. The GRangeaList store a `list` of GRanges. Each GRanges stores the raw mehylation data of one sample. The second file a numeric `vector` denoting controls and cases (a file is generates by entry in the `vector` parameters vNbSample).
saveMethylKit	a `logical`, when TRUE, for each simulations save a file contains a `list`. The length of the `list` corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a S4 `methylRawList` object. The `methylRawList` object contains two Slots: 1. treatment: A numeric `vector` denoting controls and cases. 2. .Data: A `list` of `methylRaw` objects. Each object stores the raw methylation data of one sample.
runAnalysis	a `logical`, if TRUE, two files are saved for each simulation:

- 1. The first file is the methylObj... file formated with the `methylkit` package in a S4 `methylBase` object (using the `methylKit` functions: `filterByCoverage`, `normalizeCoverage` and `unite`).
- 2. The second file contains a S4 `calculateDiffMeth` object generated using the `methylKit` functions `calculateDiffMeth` on the first file.

nbCores	a positive `integer`, the number of cores used when creating the simulated datasets. Default: 1 and always 1 for Windows.
vSeed	a `integer`, a seed used when reproducible results are needed. When a value inferior or equal to zero is given, a random integer is used. .

Value

`0` indicating that the function have been successful.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Load dataset containing methyl information
data(samplesForChrSynthetic)

## Set the output directory where files will be created
temp_dir <- "test_runOnEachSynCHR"

## Create directory
if(!dir.exists(temp_dir)) {
    dir.create(temp_dir, showWarnings = TRUE)
}
```

```
## Create 2 simulated dataset (nbSimulation = 2)
## over 3 generations (nbGenration = 3) with
## 6 cases and 6 controls (nNbsample = 6) using only one set
## of parameters (vpDiff = 0.9, vpDiffsd = 0.1, vDiff = 0.8)
methInheritSim:::runOnEachSynCHR(methData = samplesForChrSynthetic,
    nbSynCHR = 1, nbSimulation = 2, nbBlock = 10, nbCpG = 20,
    nbGeneration = 3, vNbSample = c(6), vpDiff = c(0.9), vpDiffsd = c(0.1),
    vDiff = c(0.8), vInheritance = c(0.5), propInherite = 0.3,
    rateDiff = 0.3, minRate = 0.2, propHetero = 0.5, keepDiff = FALSE,
    outputDir = temp_dir, fileID = "F1",  minReads = 10,
    maxPercReads = 99.9, meanCov = 80, context = "CpG", assembly="Rnor_5.0",
    saveGRanges = FALSE, saveMethylKit = FALSE,
    runAnalysis = FALSE, nbCores = 1, vSeed = 32)

## Delete the output directory and its content
if (dir.exists(temp_dir)) {
    unlink(temp_dir, recursive = TRUE, force = FALSE)
}
```

---

runSim   *Simulate a multigeneration methylation experiment with inheritance*

Description

Simulate a multigeneration methylation case versus control experiment with inheritance relation using a real control dataset.

The simulation can be parametrized to fit different models. The number of cases and controls, the proportion of the case affected by the treatment (penetrance), the effect of the treatment on the mean of the distribution, the proportion of sites inherited, the proportion of the differentially methylated sites from the precedent generation inherited, etc..

The function simulates a multigeneration dataset like a bisulfite sequencing experiment. The simulation includes the information about control and case for each generation. Simulation dataset are saved in multiple files created in the directory specified by the user.

Usage

```
runSim(methData, nbSynCHR = 1, nbSimulation = 10, nbBlock = 100,
  nbCpG = 50, nbGeneration = 3, vNbSample = c(3, 6), vpDiff = c(0.9),
  vpDiffsd = c(0.1), vDiff = c(0.8), vInheritance = c(0.5),
  rateDiff = 0.01, minRate = 0.01, propInherite = 0.3, propHetero = 0.5,
  keepDiff = FALSE, outputDir = NULL, fileID = "s", minReads = 10,
  maxPercReads = 99.9, meanCov = 80, context = "CpG",
  assembly = "Rnor_5.0", saveGRanges = TRUE, saveMethylKit = TRUE,
  runAnalysis = FALSE, nbCores = 1, vSeed = -1)
```

Arguments

methData   an object of class `methylBase`, the CpG information from controls (CTRL) that will be used to create the synthetic chromosome. The `methData` object can also contain information from cases but only the controls are used.

nbSynCHR	a positive `integer`, the number of distinct synthetic chromosomes that will be generated. Default: `1`.
nbSimulation	a positive `integer`, the number of simulations generated for each parameter (vNbSample, vpDiff, vDiff and vInheritance). The total number of simulation is nbSimulation * length(vNbSample) * length(vpDiff) * length(vInheritance)) Default: `10`.
nbBlock	a positive `integer`, the number of blocks used for sampling. Default: `100`.
nbCpG	a positive `integer`, the number of consecutive CpG positions used for sampling from `methInfo`. Default: `50`.
nbGeneration	a positive `integer`, the number of generations simulated. Default: `3`.
vNbSample	a `vector` of distinct positive `integer`, the number of controls (CTRL) and cases in the simulated dataset. In the simulated dataset, the number of CTRL equals the number of cases. The number of CTRL do not need to be equal to the number of Case in the real `methData` dataset. Default: `c(3, 6)`.
vpDiff	a `vector` of distinct `double` superior to `0` and inferior or equal to 1, the mean value for the proportion of samples that will have, for a specific position, differentially methylated values. It can be interpreted as the penetrance. Note that vpDiff and vpDiffsd must be the same length. Default: `c(0.9)`.
vpDiffsd	a `vector` of a non-negative `double`, the standard deviation associated to the vpDiff. Note that `vpDiff` and `vpDiffsd` must be the same length. Default: `c(0.1)`.
vDiff	a `vector` of distinct non-negative `double` included in [0,1], the proportion of C/T for a case differentially methylated that follows a beta distribution where the mean is shifted by `vDiff` from the CTRL distribution. Default: `c(0.8)`.
vInheritance	a `vector` of distinct non-negative `double` included in [0,1], the proportion of cases that inherits differentially methylated sites. Default: `c(0.5)`.
rateDiff	a positive `double` inferior to 1, the mean of the chance that a site is differentially methylated. Default: `0.01`.
minRate	a non-negative `double` inferior to 1, the minimum rate for differentially methylated sites. Default: `0.01`.
propInherite	a non-negative `double` inferior or equal to 1, the proportion of differentially methylated regions that is inherited. Default: `0.3`.
propHetero	a non-negative `double` between [0,1], the reduction of `vDiff` for the second and following generations. Default: `0.5`.
keepDiff	a `logical`, when `TRUE`, the differentially methylated sites will be the same for all simulated datasets. Datasets generated using differents parameter values from vector parameters (vpDiff, vDiff and vInheritance) wil all have the same differentially methylated sites. Default: `FALSE`.
outputDir	a string of `character` or `NULL`, the path where the files created by the function will be saved. When `NULL`, the files are saved in a directory called "outputDir" that is located in the current directory. Default: `NULL`.
fileID	a string of `character`, a identifiant that will be included in each output file name. Each output file name is composed of those elements, separated by "_":

- a type name, ex: methylGR, methylObj, etc..
- a `fileID`
- the chromosome number, a number between 1 and `nbSynCHR`
- the number of samples, a number in the `vNbSample` vector

- the mean proportion of samples that has, for a specific position, differentially methylated values, a number in the `vpDiff` vector
- the proportion of C/T for a case differentially methylated that follows a shifted beta distribution, a number in the `vDiff` vector
- the proportion of cases that inherits differentially sites, a number in the `vInheritance` vector
- the identifiant for the simulation, a number between 1 and `nbSimulation`
- the file extension ".rds"

Default: `"s"`.

`minReads`	a positive `integer`, sites and regions having lower coverage than this count are discarded. The parameter corresponds to the `lo.count` parameter in the `methylKit` package. Default: 10.
`maxPercReads`	a double between [0,100], the percentile of read counts that is going to be used as upper cutoff. Sites and regions having higher coverage than `maxPercReads` are discarded. This parameter is used for both CpG sites and tiles analysis. The parameter correspond to the `hi.perc` parameter in the `methylKit` package. Default: 99.9.
`meanCov`	a positive `integer`, the mean of the coverage at the simulated CpG sites. Default: 80.
`context`	a string of `character`, the short description of the methylation context, such as "CpG", "CpH", "CHH", etc.. Default: `"CpG"`.
`assembly`	a string of `character`, the short description of the genome assembly, such as "mm9", "hg18", etc.. Default: `"Rnor_5.0"`.
`saveGRanges`	a `logical`, when `true`, the package save two files type. The first generate for each simulation contains a `list`. The length of the `list` corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a `GRangesList`. The `GRangeaList` store a `list` of `GRanges`. Each `GRanges` stores the raw mehylation data of one sample. The second file a numeric `vector` denoting controls and cases (a file is generates by entry in the `vector` parameters `vNbSample`). Default: TRUE.
`saveMethylKit`	a `logical`, when TRUE, for each simulations save a file contains a `list`. The length of the `list` corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a S4 `methylRawList` object. The `methylRawList` object contains two Slots: 1. treatment: A numeric `vector` denoting controls and cases. 2. .Data: A `list` of `methylRaw` objects. Each object stores the raw methylation data of one sample. Default: TRUE.
`runAnalysis`	a `logical`, if TRUE, two files are saved for each simulation:

- 1. The first file is the methylObj... file formated with the `methylkit` package in a S4 `methylBase` object (using the `methylKit` functions: `filterByCoverage`, `normalizeCoverage` and `unite`).
- 2. The second file contains a S4 `calculateDiffMeth` object generated using the `methylKit` functions `calculateDiffMeth` on the first file.

Default: FALSE.

`nbCores`	a positive `integer`, the number of cores used when creating the simulated datasets. Default: 1 and always 1 for Windows.
`vSeed`	a `integer`, a seed used when reproducible results are needed. When a value inferior or equal to zero is given, a random integer is used. Default: `-1`.

Value

`0` indicating that the function have been successful.

Author(s)

Pascal Belleau, Astrid Deschenes

See Also

the vignette for detail description of the files created by the simulation.

Examples

```
## Load dataset containing methyl information
data(samplesForChrSynthetic)

## Set the output directory where files will be created
temp_dir <- "test_runSim"

## Create 2 simulated dataset (nbSimulation = 2)
## over 3 generations (nbGenration = 3) with
## 6 cases and 6 controls (nNbsample = 6) using only one set
## of parameters (vpDiff = 0.9, vpDiffsd = 0.1, vDiff = 0.8)
runSim(methData = samplesForChrSynthetic, nbSynCHR = 1, nbSimulation = 2,
    nbGeneration = 3, nbBlock = 10, nbCpG = 20, vNbSample = c(6),
    vpDiff = c(0.9), vpDiffsd = c(0.1), vDiff = c(0.8),
    vInheritance = c(0.5), rateDiff = 0.3, minRate = 0.2,
    propInherite = 0.3, propHetero = 0.5, outputDir = temp_dir,
    fileID = "F", nbCores = 1, vSeed = 32)

## Delete the output directory and its content
if (dir.exists(temp_dir)) {
    unlink(temp_dir, recursive = TRUE, force = FALSE)
}
```

---

samplesForChrSynthetic

All samples information, formated by methylKit, *in a* methylBase
format (for demo purpose).

Description

The object is a `methylBase`. There is 12 samples (6 controls and 6 cases). Each sample information is stored in a `methylRaw` object.

Usage

```
data(samplesForChrSynthetic)
```

Format

A `methylBase` object contains the information for one generation. Each sample information is stored in a `methylRaw` object. There is `methylRaw` objects (6 controls and 6 cases).

Details

This dataset can be used to test the `runSim` function.

Value

A `methylBase` contains the information for one generation. Each sample information is stored in a `methylRaw` object. There is `methylRaw` objects (6 controls and 6 cases).

See Also

- `runSim` for running a simulation analysis using methylKit info as input

Examples

```
## Loading dataset
data(samplesForChrSynthetic)

## Set the output directory where files will be created
temp_dir <- "test_samplesForChrSynthetic"

## Create 4 simulated dataset (nbSimulation)
## over 3 generations (nbGenration = 3) with
## 6 cases and 6 controls (nNbsample = 6) using only one set
## of parameters (vpDiff = 0.85, vpDiffsd = 0.1, vDiff = 0.8)
runSim(outputDir = temp_dir, fileID = "F1", nbSynCHR = 1,
    methData = samplesForChrSynthetic, nbSimulation = 4,
    nbBlock = 10, nbCpG = 20,
    nbGeneration = 3, vNbSample = c(6), vpDiff = c(0.85),
    vpDiffsd = c(0.1), vDiff = c(0.8),
    vInheritance = c(0.5), propInherite = 0.3,
    rateDiff = 0.3, minRate = 0.2, propHetero = 0.5,
    nbCores = 1, vSeed = 32)

## Delete the output directory and its content
if (dir.exists(temp_dir)) {
    unlink(temp_dir, recursive = TRUE, force = FALSE)
}
```

---

saveData   *Save data created during the simulation*

Description

Save data created during the simulation.

Usage

```
saveData(pathOut, extension, gRanges, methylData, methUnit, diffData,
  saveGRanges, saveMethylKit, runAnalysis)
```

Arguments

pathOut	a string of character, the path where the files are saved.
extension	a string of character representing the extension that will be given to the saved files.
gRanges	a list of methylRawList TODO
methylData	a list of methylRawList, the results of the normalization of the coverage.
methUnit	a list of methylBase, the results of the base filtering for all samples.
diffData	a list of methylDiff, the results of the calculation of differential methylation statistics.
saveGRanges	a logical, when true, files containing GRangeaList are saved.
saveMethylKit	a logical, when TRUE, files methylRawList object are saved.
runAnalysis	a logical, when TRUE, two files related to the analysis are saved.

Value

0 indicating that the function has been successful.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## TODO
```

---

simEachGeneration	*Simulate a multigeneration methylation experiment with inheritance*

Description

Simulate a multigeneration methylation case versus control experiment with inheritance relation using a real control dataset.

The simulation can be parametrized to fit different models. The number of cases and controls, the proportion of the case affected by the treatment (penetrance), the effect of the treatment on the mean of the distribution, the proportion of sites inherited, the proportion of the differentially methylated sites from the precedent generation inherited, etc..

The function simulates a multigeneration dataset like a bisulfite sequencing experiment. The simulation includes the information about control and case for each generation.

Usage

```
simEachGeneration(simulation, nbCtrl, nbCase, treatment, sample.id, generation,
  stateInfo, minReads, maxPercReads, context, assembly, meanCov, saveGRanges,
  saveMethylKit, runAnalysis)
```

Arguments

nbCtrl	a positive integer, the number of controls.
nbCase	a positive integer, the number of cases.
treatment	a numeric vector denoting controls and cases
sample.id	a matrix the name of each samples for each generation (row) and each case and control (column).
generation	a positive integer, the number of generations simulated.
stateInfo	a GRanges that contains the CpG (or methylated sites). The GRanges have four metadata from the real dataset:

- chrOri a numeric, the chromosome from the real dataset
- startOri a numeric, the position of the site in the real dataset
- meanCTRL a numeric, the mean of the control in the real dataset
- varCTRL a numeric, the variance of the control in the real dataset.

minReads	a positive integer, sites and regions having lower coverage than this count are discarded. The parameter corresponds to the lo.count parameter in the methylKit package.
maxPercReads	a double between [0,100], the percentile of read counts that is going to be used as upper cutoff. Sites and regions having higher coverage than maxPercReads are discarded. This parameter is used for both CpG sites and tiles analysis. The parameter correspond to the hi.perc parameter in the methylKit package.
context	a string of character, the short description of the methylation context, such as "CpG", "CpH", "CHH", etc..
assembly	a string of character, the short description of the genome assembly, such as "mm9", "hg18", etc..
meanCov	a positive integer, the mean of the coverage at the simulated CpG sites.
saveGRanges	a logical, when true, the package save two files type. The first generate for each simulation contains a list. The length of the list corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a GRangesList. The GRangeaList store a list of GRanges. Each GRanges stores the raw mehylation data of one sample. The second file a numeric vector denoting controls and cases (a file is generates by entry in the vector parameters vNbSample).
saveMethylKit	a logical, when TRUE, the package save a file contains a list. The length of the list corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a S4 methylRawList object. The methylRawList object contains two Slots: 1. treatment: A numeric vector denoting controls and cases. 2. .Data: A list of methylRaw objects. Each object stores the raw methylation data of one sample.
runAnalysis	a logical, if TRUE, two files are saved :

- 1. The first file is the methylObj... file formated with the methylkit package in a S4 methylBase object (with the methylKit functions: filterByCoverage, normalizeCoverage and unite).
- 2. The second file contains a S4 calculateDiffMeth object generated with the methylKit functions calculateDiffMeth using the first file.

Value

`0` indicating that the function has been successful.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Load dataset
data("samplesForChrSynthetic")
data("dataSimExample")

## Generate a stateInfo object using samples
stateInformation <- methInheritSim:::getSyntheticChr(methInfo =
    samplesForChrSynthetic, nbBlock = 1, nbCpG = 3)

## Generate a stateDiff and stateInherite objects with length corresponding
## to nbBlock * nbCpG from stateInformation
stateDiff   <- c(1, 0, 1)
stateInherite <- c(1, 0, 0)

## Create simulation
sim <- methInheritSim:::getSim(nbCtrl = 3, nbCase = 2,
    generation = 3, stateInfo = stateInformation, stateDiff = stateDiff,
    stateInherite = stateInherite, diffValue = 10,
    propDiff = 0.8, propDiffsd = 0.2, propInheritance = 0.8,
    propHetero = 0.1)

## TODO
methInheritSim:::simEachGeneration(simulation = sim,
nbCtrl = 3, nbCase = 2, treatment = c(0,0,0,1,1),
sample.id = dataSimExample$sample.id,
generation = 3, stateInfo = stateInformation, minReads = 10,
maxPercReads = 99, context = "Cpg", assembly = "RNOR_5.0", meanCov = 80,
saveGRanges = FALSE, saveMethylKit = FALSE, runAnalysis = FALSE)
```

---

simInheritance	*Simulate a multigenerational methylation experiment with inheritance*

Description

Simulate a multigenerational methylation case versus control experiment with inheritance relation using a real control dataset.

The simulation can be parametrized to fit different models. The number of cases and controls, the proportion of the case affected by the treatment (penetrance), the effect of the treatment on the mean of the distribution, the proportion of sites inherited, the proportion of the differentially methylated sites from the precedent generation inherited, etc..

The function simulates a multigeneration dataset like a bisulfite sequencing experiment. The simulation includes the information about control and case for each generation.

Usage

```
simInheritance(pathOut, pref, k, nbCtrl, nbCase, treatment, sample.id,
    generation, stateInfo, propDiff, propDiffsd, diffValue, propInheritance,
    rateDiff, minRate, propInherite, propHetero, minReads, maxPercReads, context,
    assembly, meanCov, diffRes, saveGRanges, saveMethylKit, runAnalysis)
```

Arguments

pathOut	a string of character or NULL, the path where the files created by the function will be saved. When NULL, the files are saved in the current directory.
pref	a string of character representing the parameters of specific simulation the string is composed of those elements, separated by "_":

- a fileID
- the chromosome number, a number between 1 and nbSynCHR
- the number of samples, a number in the vNbSample vector
- the mean proportion of samples that has, for a specific position, differentially methylated values, a number in the vpDiff vector
- the proportion of C/T for a case differentially methylated that follows a shifted beta distribution, a number in the vDiff vector
- the proportion of cases that inherits differentially sites, a number in the vInheritance vector

k	a positive integer, an ID for the current simulation.
nbCtrl	a positive integer, the number of controls.
nbCase	a positive integer, the number of cases.
treatment	a vector of integer denoting controls and cases. The vector length must correspond to the sum of cases and controls.
sample.id	a matrix the name of each samples for each generation (row) and each case and control (column).
generation	a positive integer, the number of generations simulated.
stateInfo	a GRanges that contains the CpG (or methylated sites). The GRanges have four metadata from the real dataset:

- chrOri a numeric, the chromosome from the real dataset
- startOri a numeric, the position of the site in the real dataset
- meanCTRL a numeric, the mean of the control in the real dataset
- varCTRL a numeric, the variance of the control in the real dataset.

propDiff	a double superior to 0 and inferior or equal to 1, the mean value for the proportion of samples that will have, for a specific position, differentially methylated values. It can be interpreted as the penetrance.
propDiffsd	a non-negative double, the standard deviation associated to the vpDiff. Note that vpDiff and vpDiffsd must be the same length.
diffValue	a non-negative double included in [0,1], the proportion of C/T for a case differentially methylated that follows a beta distribution where the mean is shifted by vDiff from the CTRL distribution.
propInheritance	
	a non-negative double included in [0,1], the proportion of cases that inherits differentially methylated sites.

rateDiff	a positive `double` inferior to 1, the mean of the chance that a site is differentially methylated.
minRate	a non-negative `double` inferior to 1, the minimum rate for differentially methylated sites. Default: `0.01`.
propInherite	a non-negative `double` inferior or equal to 1, the proportion of differentially methylated regions that are inherated.
propHetero	a non-negative `double` between [0,1], the reduction of `vDiff` for the second and following generations.
minReads	a positive `integer`, sites and regions having lower coverage than this count are discarded. The parameter corresponds to the `lo.count` parameter in the `methylKit` package.
maxPercReads	a `double` between [0,100], the percentile of read counts that is going to be used as upper cutoff. Sites and regions having higher coverage than `maxPercReads` are discarded. This parameter is used for both CpG sites and tiles analysis. The parameter correspond to the `hi.perc` parameter in the `methylKit` package.
context	a string of `character`, the short description of the methylation context, such as "CpG", "CpH", "CHH", etc..
assembly	a string of `character`, the short description of the genome assembly, such as "mm9", "hg18", etc..
meanCov	a positive `integer`, the mean of the coverage at the simulated CpG sites.
diffRes	a `list` with 2 entries: `stateDiff` a vector of `integer` (`0` and `1`) with length corresponding the length of `stateInfo`. The `vector` indicates, using a `1`, the positions where the CpG sites are differentially methylated. `stateInherite` a vector of `integer` (`0` and `1`) with length corresponding the length of `stateInfo`. The `vector` indicates, using a `1`, the positions where the CpG values are inherited. when is `NULL` generate a new ones with `getDiffMeth`.
saveGRanges	a `logical`, when `true`, the package save two files type. The first generate for each simulation contains a `list`. The length of the `list` corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a `GRangesList`. The `GRangeaList` store a `list` of `GRanges`. Each `GRanges` stores the raw mehylation data of one sample. The second file a numeric `vector` denoting controls and cases (a file is generates by entry in the `vector` parameters vNbSample).
saveMethylKit	a `logical`, when `TRUE`, the package save a file contains a `list`. The length of the `list` corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a S4 `methylRawList` object. The `methylRawList` object contains two Slots: 1. treatment: A numeric `vector` denoting controls and cases. 2. .Data: A `list` of `methylRaw` objects. Each object stores the raw methylation data of one sample.
runAnalysis	a `logical`, if `TRUE`, two files are saved : 1. The first file is the methylObj... file formated with the `methylkit` package in a S4 `methylBase` object (with the `methylKit` functions: `filterByCoverage`, `normalizeCoverage` and `unite`). 2. The second file contains a S4 `calculateDiffMeth` object generated with the `methylKit` functions `calculateDiffMeth` using the first file.

Value

`0` indicating that the function has been successful.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Name of the directory that will contained the generated files
temp_dir <- "test_simInheritance"

## Load dataset
data(dataSimExample)

## Generate a stateDiff object with length corresponding to
## nbBlock * nbCpG from stateInformation
stateDiff <- list()
stateDiff[["stateDiff"]] <- c(1, 0, 1)
stateDiff[["stateInherite"]] <- c(1, 0, 0)

## Simulate multigenerational methylation experiment with inheritance
methInheritSim:::simInheritance(pathOut = temp_dir,
    pref = "S1_6_0.9_0.8_0.5", k = 1, nbCtrl = 6, nbCase = 6,
    treatment = dataSimExample$treatment,
    sample.id = dataSimExample$sample.id,
    generation = 3, stateInfo = dataSimExample$stateInfo[1:3],
    propDiff = 0.9, propDiffsd = 0.1, diffValue = 0.8,
    propInheritance = 0.5, rateDiff = 0.3, minRate = 0.3,
    propInherite = 0.3, propHetero = 0.5, minReads = 10, maxPercReads = 99,
    assembly="RNOR_5.0", context="Cpg", meanCov = 40, diffRes = stateDiff,
    saveGRanges = FALSE, saveMethylKit = FALSE, runAnalysis = FALSE)

## Delete directory
if (dir.exists(temp_dir)) {
    unlink(temp_dir, recursive = TRUE, force = FALSE)
}
```

---

testIfAlreadyDone   *Test if a specific simulation has already be done.*

Description

Test if a specific simulation has already be done.

Usage

```
testIfAlreadyDone(pathOut, preference, id, saveGRanges, saveMethylKit,
  runAnalysis)
```

Arguments

pathOut	a string of `character`, the path where the files are saved.
preference	a string of `character` representing the parameters of specific simulation.
id	a positive `integer`, a ID for the current simulation.
saveGRanges	a `logical`, when `true`, files containing GRangeaList are saved.
saveMethylKit	a `logical`, when `TRUE`, files `methylRawList` object are saved.
runAnalysis	a `logical`, when `TRUE`, two files related to the analysis are saved.

Value

`logical` indicating if the simulation has already done.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Return TRUE when the specified simulation has already be done;
## otherwise, return FALSE.
methInheritSim:::testIfAlreadyDone(pathOut = ".",
preference = "S1_6_0.9_0.8_0.5", id = 33,
saveGRanges = FALSE, saveMethylKit = FALSE, runAnalysis = FALSE)
```

---

validateRunSimDoubleParameters

Parameters validation for the [runSim](#) *function. Only double parameters are validated.*

Description

Validation of all parameters needed by the public [runSim](#) function. Only double parameters are validated.

Usage

```
validateRunSimDoubleParameters(vpDiff, vpDiffsd, vDiff, vInheritance,
  propInherite, rateDiff, minRate, propHetero, maxPercReads)
```

Arguments

vpDiff	a double superior to `0` and inferior or equal to 1, the mean value for the proportion of samples that will have, for a specific position, differentially methylated values. It can be interpreted as the penetrance.
vpDiffsd	a non-negative `double`, the standard deviation associated to the `propDiff`.
vDiff	a positive double between [0,1], the proportion of C/T for a case differentially methylated follow a beta distribution where the mean is shifted of vDiff from the CTRL distribution

vInheritance	a positive double between [0,1], the proportion of cases that inherited differentially sites.
propInherite	a non-negative double inferior or equal to 1, the proportion of differentially methylated site are inherated
rateDiff	a positive double inferior to 1, the mean of the chance that a site is differentially methylated.
minRate	a non-negative double inferior to 1, the minimum rate of differentially methylated sites.
propHetero	a positive double between [0,1], the reduction of vDiff for the second and following generations.
maxPercReads	a double between [0,100], the percentile of read counts that is going to be used as upper cutoff. Bases ore regions having higher coverage than this percentile are discarded. Parameter used for both CpG sites and tiles analysis. The parameter correspond to the hi.perc parameter in the methylKit package.

Value

0 indicating that the function has been successful.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## The function returns 0 when all paramaters are valid
methInheritSim:::validateRunSimDoubleParameters(vpDiff =0.2,
vpDiffsd = 0.3, vDiff = 0.4, vInheritance = 0.2, propInherite = 0.5,
rateDiff = 0.2, minRate = 0.1, propHetero = 0.2, maxPercReads = 99.1)
```

---

validateRunSimIntegerParameters

> *Parameters validation for the* [runSim](#) *function. Only integer parameters are validated.*

Description

Validation of all parameters needed by the public [runSim](#) function. Only integer parameters are validated.

Usage

```
validateRunSimIntegerParameters(nbSynCHR, nbSimulation, nbBlock, nbCpG,
  vNbSample, nbGeneration, minReads, meanCov, nbCores, vSeed)
```

Arguments

nbSynCHR	a positive `integer`, the number of distinct synthetic chromosomes that will be generated.
nbSimulation	a positive `integer`, the number of simulations for each parameter (`vNbSample`, `vpDiff`, `vDiff` and `vInheritance`).
nbBlock	a positive `integer`, the number of blocks used for sampling.
nbCpG	a positive `integer`, the number of consecutive CpG positions used for sampling from `methInfo`.
vNbSample	a `vector` of positive `integer`, the number of methData (CTRL) and cases in the the simulation dataset. In the simulated dataset, the number of CTRL equals the number of Case. The number of CTRL do not need to be equal to the number of Case in the real dataset.
nbGeneration	a positive `integer`, the number of generations.
minReads	a positive `integer` Bases and regions having lower coverage than this count are discarded. The parameter correspond to the `lo.count` parameter in the `methylKit` package.
meanCov	a positive `integer` represent the mean of the coverage at the CpG site Default: `80`.
nbCores	a positive `integer`, the number of cores to use when creating the simulated datasets. Default: `1` and always `1` for Windows.
vSeed	a `integer`, a seed used when reproducible results are needed. When a value inferior or equal to zero is given, a random integer is used. Default: `-1`.

Value

`0` indicating that the function has been successful.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## The function returns 0 when all paramaters are valid
methInheritSim:::validateRunSimIntegerParameters(nbSynCHR = 1,
nbSimulation = 2, nbBlock = 10, nbCpG = 4, vNbSample = 10,
nbGeneration = 3, minReads = 10, meanCov = 80,
nbCores = 1, vSeed = -1)
```

---

validateRunSimLogicalParameters

> *Parameters validation for the* [runSim](#) *function. Only logical parameters are validated.*

Description

Validation of all parameters needed by the public [runSim](#) function. Only logical parameters are validated.

Usage

```
validateRunSimLogicalParameters(keepDiff, saveGRanges, saveMethylKit,
  runAnalysis)
```

Arguments

keepDiff   logical if true, the differentially methyled sites will be the same for each parameter (vpDiff, vDiff and vInheritance). Default: FALSE.

saveGRanges   a logical, when true, the package save two files type. The first generate for each simulation contains a list. The length of the list corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a GRangesList. The GRangeaList store a list of GRanges. Each GRanges stores the raw mehylation data of one sample. The second file a numeric vector denoting controls and cases (a file is generates by entry in the vector parameters vNbSample).

saveMethylKit   a logical, when TRUE, for each simulations save a file contains a list. The length of the list corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a S4 methylRawList object. The methylRawList object contains two Slots: 1. treatment: A numeric vector denoting controls and cases. 2. .Data: A list of methylRaw objects. Each object stores the raw methylation data of one sample.

runAnalysis   a logical, if TRUE, two files are saved for each simulation:

- 1. The first file is the methylObj... file formated with the methylkit package in a S4 methylBase object (with the methylKit functions: filterByCoverage, normalizeCoverage and unite).

- 2. The second file contains a S4 calculateDiffMeth object generated with the methylKit functions calculateDiffMeth on the first file.

Value

0 indicating that the function has been successful.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Load dataset
data("samplesForChrSynthetic")

## The function returns 0 when all paramaters are valid
methInheritSim:::validateRunSimLogicalParameters(keepDiff = FALSE,
saveGRanges = TRUE, saveMethylKit = FALSE, runAnalysis = FALSE)
```

validateRunSimOtherParameters

> *Parameters validation for the* runSim *function. Only parameters other than double, integer and logical are validated.*

Description

Validation of all parameters needed by the public runSim function. Only parameters other than double, integer and logical are validated.

Usage

```
validateRunSimOtherParameters(outputDir, fileID, methData, context, assembly)
```

Arguments

outputDir   a string of character or NULL, the path where the files created by the function will be saved. When NULL, the files are saved in the current directory.

fileID   a string of character, a identifiant that will be included in each output file name. Each output file name is composed of those elements, separated by "_":

- a type name, ex: methylGR, methylObj, etc..
- a fileID
- the chromosome number, a number between 1 and nbSynCHR
- the number of samples, a number in the vNbSample vector
- the mean proportion of samples that has, for a specific position, differentially methylated values, a number in the vpDiff vector
- the proportion of C/T for a case differentially methylated that follows a shifted beta distribution, a number in the vDiff vector
- the proportion of cases that inherits differentially sites, a number in the vInheritance vector
- the identifiant for the simulation, a number between 1 and nbSimulation
- the file extension ".rds"

methData   an object of class methylBase, the CpG information from controls (CTRL) that will be used to create the sythetic chromosome. The methData object can also contain information from cases but only the controls will be used.

context   a string of character, the methylation context string, ex: CpG,CpH,CHH, etc.

assembly   a string of character, the short description of the genome assembly. Ex: mm9,hg18 etc.

Value

0 indicating that the function has been successful.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Load dataset
data("samplesForChrSynthetic")

## The function returns 0 when all paramaters are valid
methInheritSim:::validateRunSimOtherParameters(
outputDir = "test", fileID = "test", methData = samplesForChrSynthetic,
context = "CpG", assembly = "Rnor_5.0")
```

---

validateRunSimParameters

Parameters validation for the runSim *function.*

Description

Validation of all parameters needed by the public runSim function.

Usage

```
validateRunSimParameters(vpDiff, vpDiffsd, vDiff, vInheritance, propInherite,
  rateDiff, minRate, propHetero, maxPercReads, nbSynCHR, nbSimulation, nbBlock,
  nbCpG, vNbSample, nbGeneration, minReads, meanCov, nbCores, vSeed, keepDiff,
  saveGRanges, saveMethylKit, runAnalysis, outputDir, fileID, methData, context,
  assembly)
```

Arguments

vpDiff	a double superior to 0 and inferior or equal to 1, the mean value for the proportion of samples that will have, for a specific position, differentially methylated values. It can be interpreted as the penetrance.
vpDiffsd	a non-negative double, the standard deviation associated to the propDiff.
vDiff	a positive double between [0,1], the proportion of C/T for a case differentially methylated follow a beta distribution where the mean is shifted of vDiff from the CTRL distribution
vInheritance	a positive double between [0,1], the proportion of cases that inherited differentially sites.
propInherite	a non-negative double inferior or equal to 1, the proportion of differentially methylated site are inherated
rateDiff	a positive double inferior to 1, the mean of the chance that a site is differentially methylated.
minRate	a non-negative double inferior to 1, the minimum rate of differentially methylated sites.
propHetero	a positive double between [0,1], the reduction of vDiff for the second and following generations.
maxPercReads	a double between [0,100], the percentile of read counts that is going to be used as upper cutoff. Bases ore regions having higher coverage than this percentile are discarded. Parameter used for both CpG sites and tiles analysis. The parameter correspond to the hi.perc parameter in the methylKit package.

nbSynCHR	a positive `integer`, the number of distinct synthetic chromosomes that will be generated.
nbSimulation	a positive `integer`, the number of simulations for each parameter (`vNbSample`, `vpDiff`, `vDiff` and `vInheritance`).
nbBlock	a positive `integer`, the number of blocks used for sampling.
nbCpG	a positive `integer`, the number of consecutive CpG positions used for sampling from `methInfo`.
vNbSample	a `vector` of positive `integer`, the number of methData (CTRL) and cases in the the simulation dataset. In the simulated dataset, the number of CTRL equals the number of Case. The number of CTRL do not need to be equal to the number of Case in the real dataset.
nbGeneration	a positive `integer`, the number of generations.
minReads	a positive `integer` Bases and regions having lower coverage than this count are discarded. The parameter correspond to the `lo.count` parameter in the `methylKit` package.
meanCov	a positive `integer` represent the mean of the coverage at the CpG site Default: 80.
nbCores	a positive `integer`, the number of cores to use when creating the simulated datasets. Default: 1 and always 1 for Windows.
vSeed	a `integer`, a seed used when reproducible results are needed. When a value inferior or equal to zero is given, a random integer is used. Default: -1.
keepDiff	`logical` if true, the differentially methyled sites will be the same for each parameter (`vpDiff`, `vDiff` and `vInheritance`). Default: FALSE.
saveGRanges	a `logical`, when true, the package save two files type. The first generate for each simulation contains a `list`. The length of the `list` corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a GRangesList. The GRangeaList store a `list` of GRanges. Each GRanges stores the raw mehylation data of one sample. The second file a numeric `vector` denoting controls and cases (a file is generates by entry in the `vector` parameters `vNbSample`).
saveMethylKit	a `logical`, when TRUE, for each simulations save a file contains a `list`. The length of the `list` corresponds to the number of generation. The generation are stored in order (first entry = first generation, second entry = second generation, etc..). All samples related to one generations are contained in a S4 `methylRawList` object. The `methylRawList` object contains two Slots: 1. treatment: A numeric `vector` denoting controls and cases. 2. .Data: A `list` of `methylRaw` objects. Each object stores the raw methylation data of one sample.
runAnalysis	a `logical`, if TRUE, two files are saved for each simulation:

- 1. The first file is the methylObj... file formated with the `methylkit` package in a S4 `methylBase` object (with the `methylKit` functions: `filterByCoverage`, `normalizeCoverage` and `unite`).
- 2. The second file contains a S4 `calculateDiffMeth` object generated with the `methylKit` functions `calculateDiffMeth` on the first file.

outputDir	a string of `character` or NULL, the path where the files created by the function will be saved. When NULL, the files are saved in the current directory.
fileID	a string of `character`, a identifiant that will be included in each output file name. Each output file name is composed of those elements, separated by "_":

- a type name, ex: methylGR, methylObj, etc..
- a fileID
- the chromosome number, a number between 1 and nbSynCHR
- the number of samples, a number in the vNbSample vector
- the mean proportion of samples that has, for a specific position, differentially methylated values, a number in the vpDiff vector
- the proportion of C/T for a case differentially methylated that follows a shifted beta distribution, a number in the vDiff vector
- the proportion of cases that inherits differentially sites, a number in the vInheritance vector
- the identifiant for the simulation, a number between 1 and nbSimulation
- the file extension ".rds"

methData an object of class methylBase, the CpG information from controls (CTRL) that will be used to create the sythetic chromosome. The methData object can also contain information from cases but only the controls will be used.

context a string of character, the methylation context string, ex: CpG,CpH,CHH, etc.

assembly a string of character, the short description of the genome assembly. Ex: mm9,hg18 etc.

Value

0 indicating that the function has been successful.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Load dataset
data("samplesForChrSynthetic")

## The function returns 0 when all paramaters are valid
methInheritSim:::validateRunSimParameters(vpDiff =0.2,
vpDiffsd = 0.3, vDiff = 0.4, vInheritance = 0.2, propInherite = 0.5,
rateDiff = 0.2, minRate = 0.1, propHetero = 0.2, maxPercReads = 99.1,
nbSynCHR = 1, nbSimulation = 2, nbBlock = 10, nbCpG = 4, vNbSample = 10,
nbGeneration = 3, minReads = 10, meanCov = 80,
nbCores = 1, vSeed = -1, keepDiff = FALSE, saveGRanges = TRUE,
saveMethylKit = FALSE, runAnalysis = FALSE, outputDir = "test",
fileID = "test", methData = samplesForChrSynthetic,
context = "CpG", assembly = "Rnor_5.0")
```

Index