Package 'methylInheritance'

March 12, 2025

Version 1.31.0

Date 2021-11-21

Title Permutation-Based Analysis associating Conserved Differentially Methylated Elements Across Multiple Generations to a Treatment Effect

Description Permutation analysis, based on Monte Carlo sampling, for testing the hypothesis that the number of conserved differentially methylated elements, between several generations, is associated to an effect inherited from a treatment and that stochastic effect can be dismissed.

Depends R (>= 3.5)

Imports methylKit, BiocParallel, GenomicRanges, IRanges, S4Vectors, methods, parallel, ggplot2, gridExtra, rebus

Suggests BiocStyle, BiocGenerics, knitr, rmarkdown, RUnit, methInheritSim

Encoding UTF-8

License Artistic-2.0

URL https://github.com/adeschen/methylInheritance

BugReports https://github.com/adeschen/methylInheritance/issues

VignetteBuilder knitr

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

RoxygenNote 7.1.1

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

git_branch devel

git_last_commit 9150e3e

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-03-12

Author Astrid Deschênes [cre, aut] (ORCID:
<https://orcid.org/0000-0001-7846-6749>),
Pascal Belleau [aut] (ORCID: <https://orcid.org/0000-0002-0802-1071>),
Arnaud Droit [aut]

Maintainer Astrid Deschênes <adeschen@hotmail.com>

Contents

`methylInheritance-package`

methylInheritance: Permutation-Based Analysis associating Conserved Differentially Methylated Elements from One Generation to the Next to a Treatment Effect

---

Description

This package does a permutation analysis, based on Monte Carlo sampling, for testing the hypothesis that the number of conserved differentially methylated elements (sites or tiles), between several generations, is associated to an effect inherited from a treatment and that stochastic effect can be dismissed.

Author(s)

Astrid Deschênes, Pascal Belleau and Arnaud Droit

Maintainer: Astrid Deschenes <adeschen@hotmail.com>

See Also

- `runPermutation` for running a permutation analysis, and optionally an observation analysis, on a specified multi-generational dataset
- `runObservation` for running an observation analysis on a specified multi-generational dataset

---

`calculateSignificantLevel`
Calculate significant level for hypo and hyper conserved elements

---

Description

Calculate significant level for hypo and hyper conserved elements using permutation results as well as observed results

Usage

```
calculateSignificantLevel(formatForGraphDataFrame)
```

Arguments

`formatForGraphDataFrame`
: a `data.frame` containing the observation results (using real data) and the permutation results (using shuffled data). Both hyper and hypo differentially conserved methylation results must be present. The `data.frame` must have 3 columns : "TYPE", "RESULT" and "SOURCE". The "TYPE" can be either "HYPER" or "HYPO". The "RESULT" is the number of conserved differentially elements. The "SOURCE" can be either "OBSERVATION" or "PERMUTATION".

Value

a list containing two elements:

- `HYPER` a `double`, the significant level for the hyper differentially methylated conserved elements
- `HYPO` a `double`, the significant level for the hypo differentially methylated conserved elements

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Loading dataset containing all results
data(methylInheritanceResults)

## Extract information for the intersection between conserved differentially
## methylated sites (type = sites) between the intersection of 2
## generations (inter = i2): F2 and F3 (position = 2)
info <- extractInfo(allResults = methylInheritanceResults,
    type = "sites", inter="i2", 2)

## Create graph
methylInheritance:::calculateSignificantLevel(info)
```

`createDataStructure` *Extract the number of conserved differentially methylated elements in* `GRanges`.

Description

Extract the number of conserved differentially methylated elements in `GRanges`. Each `GRanges` is the result of one intersection between two or more consecutive generations for one analysis done on all generations. The hypo and hyper differentially methylated elements are counted separatly.

Usage

```
createDataStructure(interGenerationGR)
```

Arguments

`interGenerationGR`

a `list` that contains the information for all differentially methylated analysis done on each generation present in the initial dataset. The `list` must contain the following elements:

- `i2` a `list` of `GRanges` Each `GRanges` represents the intersection of analysis results between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc.. The number of entries depends of the number of generations.
- `iAll` a `list` of `GRanges`. Each `GRanges` represents the intersection fo the analysis results between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc..The number of entries depends of the number of generations.

Value

a `list` containing the following elements:

- `i2` a `list` containing:
  - `HYPER` a `list` of `integer`, the number of conserved hyper differentially methylated sites between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc..
  - `HYPO` a `list` of `integer`, the number of conserved hypo differentially methylated sites between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc..
- `iAll` a `list` containing:
  - `HYPER` a `list` of `integer`, the number of conserved hyper differentially methylated sites between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc..The number of entries depends of the number of generations.
  - `HYPO` a `list` of `integer`, the number of conserved hypo differentially methylated sites between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc..The number of entries depends of the number of generations.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Get the name of the directory where the file is stored
filesDir <- system.file("extdata", "TEST", package="methylInheritance")

## Load file containing results from a observation analysis
obsResults <- readRDS(file = paste0(filesDir,
    "/SITES/SITES_observed_results.RDS"))

## Create data structure using information form the observation analysis
formatedResults <- methylInheritance:::createDataStructure(obsResults)
```

---

createOutputDir *Create directories that will contained the results of the permutations in RDS format*

Description

Create directories that will contained the results of the permutations in RDS format.

Usage

```
createOutputDir(
  outputDir,
  doingSites = TRUE,
  doingTiles = FALSE,
  saveInfoByGeneration
)
```

Arguments

outputDir	a string of character, the name of the main directory to be created.
doingSites	a logical, a directory consecrated to contain the results of the permutation analysis for sites is created when doingSites = TRUE. Default: TRUE.
doingTiles	a logical, a directory consecrated to contain the results of the permutation analysis for tiles is created when doingTiles = TRUE. Default: FALSE.
saveInfoByGeneration	a logical, when TRUE, the information about differentially methylated sites and tiles for each generation is saved in a RDS file. The information is saved in a different file for each permutation.

Value

0 when all directories are created without problem.

Author(s)

Astrid Deschenes

Examples

```
## Create an output directory for SITES only
methylInheritance:::createOutputDir(outputDir = "testSites",
    doingSites = TRUE, doingTiles = FALSE, saveInfoByGeneration = TRUE)
```

---

demoForTransgenerationalAnalysis

The methylation information from samples over three generations. Information for each generation is stored in a methylRawList *format (for demo purpose).*

Description

The object is a list with 3 entries. Each entry corresponds to the information for one generation (first entry = first generation, etc..) stored in a methylRawList object. There are 12 samples (6 controls and 6 cases) for each generation. Each sample information is stored in a methylRaw object.

Usage

```
data(demoForTransgenerationalAnalysis)
```

Format

A `list` containing three `methylRawList` objects. Each `methylRawList` contains the information for one generation (first entry = first generation, etc..). Each sample information is stored in a `methylRaw` object. There is `methylRaw` objects (6 controls and 6 cases) in each generation.

Details

This dataset can be used to test `runPermutation` and `runObservation` functions.

Value

A `list` containing three `methylRawList` objects. Each `methylRawList` contains the information for one generation (first entry = first generation, etc..). Each sample information is stored in a `methylRaw` object. There is `methylRaw` objects (6 controls and 6 cases) in each generation.

See Also

- `runPermutation` for running a permutation analysis, and optionally an observation analysis, using multi-generational dataset
- `runObservation` for running an observation analysis using methylKit info entry

Examples

```
## Loading dataset
data(demoForTransgenerationalAnalysis)

## Run a permutation analysis
runObservation(methylKitData = demoForTransgenerationalAnalysis,
    outputDir = "test_demo", type = "tiles", vSeed = 2001)

## Get results
result <- loadAllRDSResults(analysisResultsDir = "test_demo",
    permutationResultsDir = NULL, doingSites = FALSE,
    doingTiles = TRUE)

## Remove result directory
if (dir.exists("test_demo")) {
    unlink("test_demo", recursive = TRUE)
}
```

`extractInfo` *Extract the information specific to a subsection of the permutation analysis*

Description

Extract the information specific to a subsection of the permutation analysis. The extracted information will be specific to one type of differential methylation analysis (tiles or sites), to one type of intersection (two consecutive generation or more) and to one specific group of generations.

Usage

```
extractInfo(
  allResults,
  type = c("sites", "tiles"),
  inter = c("i2", "iAll"),
  position = 1
)
```

Arguments

`allResults`	a `list` of class `methylInheritanceAllResults` as created by the `runPermutation` function. The `list` must contain two entries : `"PERMUTATION"` and `"OBSERVATION"`. The `"PERMUTATION"` `list` must contain all results from all permutations while the `"OBSERVATION"` `list` must contain the result obtained with the observed dataset (not shuffled).
`type`	One of the `"sites"` or `"tiles"` strings. Specifies the type of differentially methylated elements should be returned. For retrieving differentially methylated bases `type = "sites"`; for differentially methylated regions `type = "tiles"`. Default: `"sites"`.
`inter`	One of the `"i2"` or `"iAll"` strings. Specifies the type of intersection should be returned. For retrieving intersection results between two consecutive generations `inter = "i2"`; for intersection results between three generations or more `inter = "iAll"`. Default: `"i2"`.
`position`	a positive `integer`, the position in the `list` where the information will be extracted. Default=`1`.

Value

a `data.frame` containing the observation results (using real data) and the permutation results (using shuffled data). Both hyper and hypo differentially conserved methylation results are present.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Get the name of the directory where files are stored
filesDir <- system.file("extdata", "TEST", package="methylInheritance")

## Load information from files
results <- loadAllRDSResults(analysisResultsDir = filesDir,
    permutationResultsDir = filesDir, doingSites = TRUE, doingTiles = TRUE)

## Extract information for the intersection between conserved differentially
## methylated sites (type = sites) between the intersection of 2
## generations (inter = i2): F1 and F2 (position = 1)
info <- extractInfo(allResults = results, type = "sites", inter="i2", 1)
```

`formatInputMethylData` *Permute dataset*

Description

Permute dataset and format it to be ready for an analysis

Usage

```
formatInputMethylData(methylKitData)
```

Arguments

`methylKitData`	a `list` of `methylRawList` entries. Each `methylRawList` entry must contain all the `methylRaw` entries related to one generation (first entry = first generation, second entry = second generation, etc..). The number of generations must correspond to the number of entries in the `methylKitData`. At least 2 generations must be present to make a permutation analysis. More information can be found in the methylKit package.

Value

a `list` of `methylRawList` entries.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

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

methylInheritance:::formatInputMethylData(samplesForTransgenerationalAnalysis)
```

getGRangesFromMethylDiff *Transform results from a CpG site or region analysis done on mutliple generations into a* list *of* GRanges *objects*

Description

Transform a list of methylDiff objects into a list of GRanges objects. Each methylDiff object represent a CpG site or region analysis done on one generation.

Usage

```
getGRangesFromMethylDiff(
  methDiff,
  pDiff,
  qvalue,
  type = c("all", "hyper", "hypo")
)
```

Arguments

methDiff	a list of S4 methylDiff class objects, each entry of the list represents the differentially methylated results for one generation (first entry = first genertation, second entry = second generation, etc..). Each methylDiff object holds statistics and locations for differentially methylated regions/bases.
pDiff	a positive double between 0 and 100, the cutoff for absolute value of methylation percentage change between test and control.
qvalue	a positive double inferior to 1, the cutoff for qvalue of differential methylation statistic.
type	One of the "hyper","hypo" or "all" strings, the string specifies what type of differentially methylated bases/tiles should be treated For retrieving hyper-methylated tiles/sites type = "hyper"; for hypo-methylated type = "hypo". Default: "all".

Value

a list of GRanges objects, each entry of the list represents the differentially methylated results for one generation (first entry = first genertation, second entry = second generation, etc..). Each GRanges object holds statistics for differentially methylated regions/bases.

Author(s)

Pascal Belleau

Examples

```
## Load permutation results on sites
permutationResultsFile <- system.file("extdata",
    "permutationResultsForSites.RDS", package="methylInheritance")
permutationResults <- readRDS(permutationResultsFile)

## Transform result to GRanges
resultsGR <- methylInheritance:::getGRangesFromMethylDiff(methDiff =
    permutationResults, pDiff = 10, qvalue = 0.01, type = "hyper")
```

`interGeneration` *Calculate the intersection of the differentially methylated results for two or more consercutive generations*

Description

Calculate the intersection of the differentially methylated results for two or more consercutive generations using a `list` of `GRanges` where each entry represents the results for one generation.

Usage

```
interGeneration(resultAllGenGR)
```

Arguments

`resultAllGenGR`	a `list` of `GRanges` as created by the `getGRangesFromMethylDiff` function. Each entry of the `list` represents the differentially methylated results for one generation (first entry = first genertation, second entry = second generation, etc..). Each `GRanges` object holds statistics for differentially methylated regions/bases.

Value

a `list` containing the following elements:

- `i2` a `list` of `GRanges` Each `GRanges` represents the intersection of analysis results between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc.. The number of entries depends of the number of generations.
- `iAll` a `list` of `GRanges`. Each `GRanges` represents the intersection fo the analysis results between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc..The number of entries depends of the number of generations.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Load permutation results on sites
permutationResultsFile <- system.file("extdata",
    "permutationResultsForSites.RDS", package="methylInheritance")
permutationResults <- readRDS(permutationResultsFile)

## Transform result to GRanges
resultsGR <- methylInheritance:::getGRangesFromMethylDiff(methDiff =
    permutationResults, pDiff = 10, qvalue = 0.01, type = "hyper")

## Extract inter generational conserved sites
conservedSitesGR <- methylInheritance:::interGeneration(resultsGR)
```

---

isInterGenerationResults

Verify if a specific file containing intergenerational results exists or not.

Description

Verify if a specific file containing intergenerational results exists or not.

Usage

```
isInterGenerationResults(outputDir, permutationID, type = c("sites", "tiles"))
```

Arguments

outputDir	a string of character, the name of the directory that will contain the results of the permutation. The name should end with a slash. The directory should already exists.
permutationID	an integer, the identifiant of the permutation. When the permutationID = 0, the results are considered as the observed results and are saved in a file with the "_observed_results.RDS" extension. When the permutationID != 0, the results are considered as permutation results and are saved in a file with the "_permutation_permutationID.RDS" extension.
type	One of the "sites" or "tiles" strings. Specifies the type of differentially methylated elements should be saved. Default: "sites".

Value

TRUE when file present; otherwise FALSE.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Get the name of the directory where the file is stored
filesDir <- system.file("extdata", "TEST", package="methylInheritance")

## Verify that DMS intergenerational results for the observed data exists
methylInheritance:::isInterGenerationResults(outputDir =
    paste0(filesDir, "/"), 0, "sites")
```

`loadAllRDSResults` *Load all RDS files created by the permutation and observation analysis*

Description

Load all RDS files created by the permutation and observation analysis. The function returns an object of `class` "methylInheritanceAllResults" that holds all the pertinent information.

Usage

```
loadAllRDSResults(
  analysisResultsDir,
  permutationResultsDir,
  doingSites = TRUE,
  doingTiles = FALSE,
  maxID = NA
)
```

Arguments

`analysisResultsDir`
: a `character` string, the path to the directory that contains the analysis results. The path can be the same as for the `permutationResultsDir` parameter. When `NULL`, the observation results are not loaded. Default = `NULL`.

`permutationResultsDir`
: a `character` string, the path to the directory that contains the permutation results. The path can be the same as for the `analysisResultsDir` parameter. When `NULL`, the permutation results are not loaded. Default = `NULL`.

`doingSites`
: a `logical`, the data related to differentially methylated sites are loaded when `doingSites` = `TRUE`. Default: `TRUE`.

`doingTiles`
: a `logical`, the data related to differentially methylated tiles are loaded when `doingTiles` = `TRUE`. Default: `TRUE`.

`maxID`
: `NA` or a positive `integer`, the maximum identification number of the permutation files to be loaded. When `NA`, all files present in the directory are loaded. Default: `NA`.

Value

a `list` of class `methylInheritanceAllResults` containing the result of the observation analysis as well as the results of all the permutations.

Author(s)

Astrid Deschenes, Pascal Belleau

See Also

`mergePermutationAndObservation` for detail description, in the Value section, of the `methylInheritanceAllResults` object.

Examples

```
## Get the name of the directory where files are stored
filesDir <- system.file("extdata", "TEST", package="methylInheritance")

## Load information from files
results <- loadAllRDSResults(analysisResultsDir = filesDir,
    permutationResultsDir = filesDir, doingSites = TRUE, doingTiles = TRUE)

## Print the observation results
results

## Access the results for the first permutation only for sites
results$PERMUTATION[[1]]$SITES
```

---

loadConvergenceData *Load convergence information from RDS files*

Description

Load convergence information from RDS files.

Usage

```
loadConvergenceData(
  analysisResultsDir,
  permutationResultsDir,
  type = c("sites", "tiles"),
  inter = c("i2", "iAll"),
  position,
  by = 100
)
```

Arguments

analysisResultsDir	a character string, the path to the directory that contains the analysis results. The path can be the same as for the permutatioNResultsDir parameter.
permutationResultsDir	a character string, the path to the directory that contains the permutation results. The path can be the same as for the analysisResultsDir parameter.
type	One of the "sites" or "tiles" strings. Specifies the type of differentially methylated elements should be returned. For retrieving differentially methylated bases type = "sites"; for differentially methylated regions type = "tiles". Default: "sites".
inter	One of the "i2" or "iAll" strings. Specifies the type of intersection should be returned. For retrieving intersection results between two consecutive generations inter = "i2"; for intersection results between three generations or more inter = "iAll". Default: "i2".
position	a positive integer, the position in the list where the information will be extracted.
by	a integer, the increment of the number of permutations where the significant level is tested. Default: 100.

Value

a graph showing the evolution of the significant level with the number of permutations

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Get the name of the directory where files are stored
filesDir <- system.file("extdata", "TEST", package="methylInheritance")

## Load convergence information
results <- loadConvergenceData(analysisResultsDir = filesDir,
    permutationResultsDir = filesDir, type="sites", inter="i2", position=1,
    by=1)
```

---

mergePermutationAndObservation

Merge the permutation results with the observation results.

Description

Merge the permutation results with the observation results. The merging is only needed when permutation and observation have been processed separately. The returned value is a methylInheritanceAllResults object that can be used by the extractInfo function.

Usage

```
mergePermutationAndObservation(permutationResults, observationResults)
```

Arguments

`permutationResults`
: a `list` with 1 entry called `PERMUTATION`. The `PERMUTATION` entry is a `list` with a number of entries corresponding to the number of permutations that have been processed. Each entry contains the result of one permutation.

`observationResults`
: a `list` with 1 entry called `OBSERVATION`. The `OBSERVATION` entry is a `list` containing the result obtained with the observed dataset (not shuffled).

Value

a `list` of class `methylInheritanceAllResults` with 2 entries. The 2 entries are:

- `PERMUTATION` `list` with a number of entries corresponding to the number of permutations that have been processed. Each entry contains the result of one permutation.The elements in each entry are:
  - `SITES` Only present when a sites analysis has been achieved, a `list` containing:
    - `i2` a `list` containing:
      - `HYPER` a `list` of `integer`, the number of conserved hyper differentially methylated sites between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc.
      - `HYPO` a `list` of `integer`, the number of conserved hypo differentially methylated sites between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc.
    - `iAll` a `list` containing:
      - `HYPER` a `list` of `integer`, the number of conserved hyper differentially methylated sites between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc.The number of entries depends on the number of generations.
      - `HYPO` a `list` of `integer`, the number of conserved hypo differentially methylated sites between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc. The number of entries depends on the number of generations.
  - `TILES` Only present when a tiles analysis has been achieved, a `list` containing:
    - `i2` a `list` containing:
      - `HYPER` a `list` of `integer`, the number of conserved hyper differentially methylated positions between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc.

        - HYPO a list of integer, the number of conserved hypo differentially methylated positions between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc.
      * iAll a list containing:
        - HYPER a list of integer, the number of conserved hyper differentially methylated positions between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc. The number of entries depends on the number of generations.
        - HYPO a list of integer, the number of conserved hypo differentially methylated positions between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc.The number of entries depends on the number of generations.
- OBSERVATION a list containing the result obtained with the observed dataset (not shuffled). The elements are:
  - SITES Only present when a sites analysis has been achieved, a list containing:
    * i2 a list containing:
      - HYPER a list of integer, the number of conserved hyper differentially methylated sites between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc.
      - HYPO a list of integer, the number of conserved hypo differentially methylated sites between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc.
    * iAll a list containing:
      - HYPER a list of integer, the number of conserved hyper differentially methylated sites between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc.The number of entries depends on the number of generations.
      - HYPO a list of integer, the number of conserved hypo differentially methylated sites between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc. The number of entries depends on the number of generations.
  - TILES Only present when a tiles analysis has been achieved, a list containing:
    * i2 a list containing:
      - HYPER a list of integer, the number of conserved hyper differentially methylated positions between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc.
      - HYPO a list of integer, the number of conserved hypo differentially methylated positions between two consecutive generations.The first element represents the in-

tersection of the first and second generations; the second element, the intersection of the second and third generations; etc.

* `iAll` a `list` containing:
  - `HYPER` a `list` of `integer`, the number of conserved hyper differentially methylated positions between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc. The number of entries depends on the number of generations.
  - `HYPO` a `list` of `integer`, the number of conserved hypo differentially methylated positions between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc.The number of entries depends on the number of generations.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Create a observation result
observed <- list()
observed[["OBSERVATION"]] <- list()
observed[["OBSERVATION"]][["SITES"]] <- list()
observed[["OBSERVATION"]][["SITES"]][["i2"]] <- list(HYPER = list(11, 10),
    HYPO = list(13, 12))
observed[["OBSERVATION"]][["SITES"]][["iAll"]] <- list(HYPER = list(1),
    HYPO = list(3))

## Create a permutation result containing only 1 permutation result
## Real perumtations results would have more entries
permutated <- list()
permutated[["PERMUTATION"]] <- list()
permutated[["PERMUTATION"]][[1]] <- list()
permutated[["PERMUTATION"]][[1]][["SITES"]] <- list()
permutated[["PERMUTATION"]][[1]][["SITES"]][["i2"]] <- list(HYPER =
    list(11, 12), HYPO = list(8, 11))
permutated[["PERMUTATION"]][[1]][["SITES"]][["iAll"]] <- list(HYPER =
    list(0), HYPO = list(1))

## Merge permutation and observation results
mergePermutationAndObservation(permutationResults = permutated,
    observationResults = observed)
```

`methylInheritanceResults`

All observed and permutation results formatted in a `methylInheritanceResults` *class (for demo purpose).*

Description

The object is a `list` with 2 entries: "OBSERVATION" and "PERMUTATION".

Usage

```
data(methylInheritanceResults)
```

Format

a `list` of class `methylInheritanceAllResults` containing the following elements:

- `OBSERVATION` a `list` containing:
  - `SITES` a `list` containing:
    - `i2` a `list` containing:
      - `HYPER` a `list` of `integer` with 2 entries, the number of conserved hyper differentially methylated sites between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
      - `HYPO` a `list` of `integer` with 2 entries, the number of conserved hypo differentially methylated sites between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
    - `iAll` a `list` containing:
      - `HYPER` a `list` of `integer` with 1 entry, the number of conserved hyper differentially methylated sites between the three consecutive generations.
      - `HYPO` a `list` of `integer` with 1 entry, the number of conserved hypo differentially methylated sites between the three consecutive generations.
  - `TILES` a `list` containing:
    - `i2` a `list` containing:
      - `HYPER` a `list` of `integer` with 2 entries, the number of conserved hyper differentially methylated positions between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
      - `HYPO` a `list` of `integer` with 2 entries, the number of conserved hypo differentially methylated positions between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
    - `iAll` a `list` containing:
      - `HYPER` a `list` of `integer` with 1 entry, the number of conserved hyper differentially methylated positions between the three consecutive generations.
      - `HYPO` a `list` of `integer` with 1 entry, the number of conserved hypo differentially methylated positions between the three consecutive generations.
- `PERMUTATION` a `list` containing `nbrPermutations` entries. Each entry is a `list` containing:
  - `SITES` a `list` containing:
    - `i2` a `list` containing:

        - HYPER a list of integer with 2 entries, the number of conserved hyper differentially methylated sites between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
        - HYPO a list of integer with 2 entries, the number of conserved hypo differentially methylated sites between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
      * iAll a list containing:
        - HYPER a list of integer with 1 entry, the number of conserved hyper differentially methylated sites between the three consecutive generations.
        - HYPO a list of integer with 1 entry, the number of conserved hypo differentially methylated sites between the three consecutive generations.
  - TILES a list containing:
      * i2 a list containing:
        - HYPER a list of integer with 2 entries, the number of conserved hyper differentially methylated positions between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
        - HYPO a list of integer with 2 entries, the number of conserved hypo differentially methylated positions between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
      * iAll a list containing:
        - HYPER a list of integer with 1 entry, the number of conserved hyper differentially methylated positions between the three consecutive generations.
        - HYPO a list of integer with 1 entry, the number of conserved hypo differentially methylated positions between the three consecutive generations.

Details

This dataset can be used to test the extractInfo function.The extracted information can be used to calculate the significant level or to create a graph.

Value

a list of class methylInheritanceAllResults containing the following elements:

- OBSERVATION a list containing:
  - SITES a list containing:
      * i2 a list containing:
        - HYPER a list of integer with 2 entries, the number of conserved hyper differentially methylated sites between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.

        - HYPO a list of integer with 2 entries, the number of conserved hypo differentially methylated sites between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
      * iAll a list containing:
        - HYPER a list of integer with 1 entry, the number of conserved hyper differentially methylated sites between the three consecutive generations.
        - HYPO a list of integer with 1 entry, the number of conserved hypo differentially methylated sites between the three consecutive generations.
    - TILES a list containing:
      * i2 a list containing:
        - HYPER a list of integer with 2 entries, the number of conserved hyper differentially methylated positions between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
        - HYPO a list of integer with 2 entries, the number of conserved hypo differentially methylated positions between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
      * iAll a list containing:
        - HYPER a list of integer with 1 entry, the number of conserved hyper differentially methylated positions between the three consecutive generations.
        - HYPO a list of integer with 1 entry, the number of conserved hypo differentially methylated positions between the three consecutive generations.
- PERMUTATION a list containing a number of entries corresponding to the number of permutations that have been produced. Each entry is a list containing:
    - SITES a list containing:
      * i2 a list containing:
        - HYPER a list of integer with 2 entries, the number of conserved hyper differentially methylated sites between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
        - HYPO a list of integer with 2 entries, the number of conserved hypo differentially methylated sites between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.
      * iAll a list containing:
        - HYPER a list of integer with 1 entry, the number of conserved hyper differentially methylated sites between the three consecutive generations.
        - HYPO a list of integer with 1 entry, the number of conserved hypo differentially methylated sites between the three consecutive generations.
    - TILES a list containing:
      * i2 a list containing:
        - HYPER a list of integer with 2 entries, the number of conserved hyper differentially methylated positions between two consecutive generations. The first element

represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.

- HYPO a list of integer with 2 entries, the number of conserved hypo differentially methylated positions between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations.

* iAll a list containing:
  - HYPER a list of integer with 1 entry, the number of conserved hyper differentially methylated positions between the three consecutive generations.
  - HYPO a list of integer with 1 entry, the number of conserved hypo differentially methylated positions between the three consecutive generations.

See Also

- extractInfo for extracting the information specific to a subsection of the permutation analysis

Examples

```
## Loading dataset containing all results
data(methylInheritanceResults)

## Extract information for the intersection between conserved differentially
## methylated sites (type = sites) between the intersection of 2
## generations (inter = i2): F1 and F2 (position = 1)
extractInfo(allResults = methylInheritanceResults,
    type = "sites", inter="i2", 1)
```

---

plotConvergenceGraph *Generate a graph showing the convergence for a permutation analysis*

Description

Generate a graph showing the convergence for a permutation analysis using observed and permuted results.

Usage

```
plotConvergenceGraph(dataFrameConvergence)
```

Arguments

dataFrameConvergence
: a data.frame containing the significant levels at different number of cycles (total number of permuted data analysed). The data.frame must have 6 columns : "NBR_PERMUTATIONS", "ELEMENT". "ANALYSIS", "POSITION", "TYPE" and "SIGNIFICANT_LEVEL". The "ELEMENT" can be either "SITES" or "TILES". The "TYPE" can be either "HYPER" or "HYPO".

Value

a ggplot object.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Get the name of the directory where files are stored
filesDir <- system.file("extdata", "TEST", package="methylInheritance")

## Extract convergenc information for F1 and F2 and F3
data <- loadConvergenceData(analysisResultsDir = filesDir,
    permutationResultsDir = filesDir, type = "sites", inter = "iAll",
    position = 1, by = 1)

## Create convergence graph
plotConvergenceGraph(data)
```

---

plotGraph *Generate a graph for a permutation analysis*

Description

Generate a graph for a permutation analysis using observed and shuffled results.

Usage

```
plotGraph(formatForGraphDataFrame)
```

Arguments

formatForGraphDataFrame

a data.frame containing the observation results (using real data) and the permutation results (using shuffled data). Both hyper and hypo differentially conserved methylation results must be present. The data.frame must have 3 columns : "TYPE", "RESULT" and "SOURCE". The "TYPE" can be either "HYPER" or "HYPO". The "RESULT" is the number of conserved differentially elements. The "SOURCE" can be either "OBSERVATION" or "PERMUTATION".

Value

a graph showing the permutation analysis results

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Loading dataset containing all results
data(methylInheritanceResults)

## Extract information for the intersection between conserved differentially
## methylated sites (type = sites) between the intersection of 2
## generations (inter = i2): F2 and F3 (position = 2)
info <- extractInfo(allResults = methylInheritanceResults,
    type = "sites", inter="i2", 2)

## Create graph
plotGraph(info)
```

---

print.methylInheritanceAllResults *Print a* methylInheritanceAllResults *object*

Description

Print a methylInheritanceAllResults object

Usage

```
## S3 method for class 'methylInheritanceAllResults'
print(x, ...)
```

Arguments

x	the output object from mergePermutationAndObservation function, runPermutationUsingRDSFile function (when runObservationAnalysis = TRUE and runPermutationUsingMethylKitInfo function (when runObservationAnalysis = TRUE to be printed
...	arguments passed to or from other methods

Value

an object of class methylInheritanceAllResults

Examples

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

## Print dataset
print(methylInheritanceResults)
```

`readInterGenerationResults` *Read and return intergenerational results contained in a RDS file*

Description

Read and return intergenerational results contained in a RDS file

Usage

```
readInterGenerationResults(
  outputDir,
  permutationID,
  type = c("sites", "tiles")
)
```

Arguments

`outputDir`	a string of `character`, the name of the directory that will contain the results of the permutation. The name should end with a slash. The directory should already exists.
`permutationID`	an `integer`, the identifiant of the permutation. When the `permutationID = 0`, the results are considered as the observed results and are saved in a file with the "_observed_results.RDS" extension. When the `permutationID != 0`, the results are considered as permutation results and are saved in a file with the "_permutation_permutationID.RDS" extension.
`type`	One of the `"sites"` or `"tiles"` strings. Specifies the type of differentially methylated elements should be saved. Default: `"sites"`.

Value

a `list` containing the intergenerational results for the specified permutation.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Get the name of the directory where the file is stored
filesDir <- system.file("extdata", "TEST", package="methylInheritance")

## Read DMS intergenerational results for the observed data
methylInheritance:::readInterGenerationResults(outputDir =
    paste0(filesDir, "/"), 0, "sites")
```

`runObservation` *Run a differential methylation analysis on multi-generational dataset*

Description

Run a differential methylation analysis on each generation present in a dataset. The number of conserved differentially methylated elements (sites, tile or both) between generations is them calculated. The methylKit package is used to identify the differentially methylated elements.

The multi-generational dataset or the name of the RDS file that contains the dataset can be used as input.

The results can also be saved in RDS file (optional).

Usage

```
runObservation(
  methylKitData,
  type = c("both", "sites", "tiles"),
  outputDir = "output",
  nbrCoresDiffMeth = 1,
  minReads = 10,
  minMethDiff = 10,
  qvalue = 0.01,
  maxPercReads = 99.9,
  destrand = FALSE,
  minCovBasesForTiles = 0,
  tileSize = 1000,
  stepSize = 1000,
  vSeed = -1,
  restartCalculation = FALSE,
  saveInfoByGeneration = FALSE
)
```

Arguments

`methylKitData`	a `list` of `methylRawList` entries or the name of the RDS file containing the list. Each `methylRawList` contains all the `methylRaw` entries related to one generation (first entry = first generation, second entry = second generation, etc..). The number of generations must correspond to the number of entries in the `methylKitData`.At least 2 generations must be present to calculate the conserved elements. More information can be found in the methylKit package.
`type`	One of the "sites","tiles" or "both" strings. Specifies the type of differentially methylated elements should be returned. For retrieving differentially methylated bases type="sites"; for differentially methylated regions type="tiles". Default: "both".
`outputDir`	a string, the name of the directory that will contain the results of the analysis. If the directory does not exist, it will be created. Default: `"output"`.

nbrCoresDiffMeth
: a positive integer, the number of cores to use for parallel differential methylation calculations.The parameter is used for both sites and tiles analysis. The parameter corresponds to the num.cores parameter in the package methylKit. Default: 1 and always 1 for Windows.

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 package methylKit.

minMethDiff
: a positive double between [0,100], the absolute value of methylation percentage change between cases and controls. The parameter corresponds to the difference parameter in the methylKit package. Default: 10.

qvalue
: a positive double between [0,1], the cutoff for qvalue of differential methylation statistics. Default: 0.01.

maxPercReads
: a double between [0,100], the percentile of read counts that is going to be used as an upper cutoff. Bases or regions having higher coverage than this percentile are discarded. The parameter is used for both CpG sites and tiles analysis. The parameter corresponds to the hi.perc parameter in the package methylKit. Default: 99.9.

destrand
: a logical, when TRUE will merge reads on both strands of a CpG dinucleotide to provide better coverage. Only advised when looking at CpG methylation. Parameter used for both CpG sites and tiles analysis. Default: FALSE.

minCovBasesForTiles
: a non-negative integer, the minimum number of bases to be covered in a given tiling window. The parameter corresponds to the cov.bases parameter in the package methylKit. Only used when doingTiles = TRUE. Default: 0.

tileSize
: a positive integer, the size of the tiling window. The parameter corresponds to the win.size parameter in the package methylKit. Only used when doingTiles = TRUE. Default: 1000.

stepSize
: a positive integer, the step size of tiling windows. The parameter corresponds to the stepSize parameter in the package methylKit. Only used when doingTiles = TRUE. Default: 1000.

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.

restartCalculation
: a logical, when TRUE, only permutations that don't have a RDS result final are run. Useful to restart a permutation analysis that has been interrupted. Beware that the parameters have to be identical except for this one.

saveInfoByGeneration
: a logical, when TRUE, the information about differentially methylated sites and tiles for each generation is saved in a RDS file. The files are saved in the directory specified by the outputDir parameter.

Value

0.

Author(s)

Astrid Deschenes, Pascal Belleau

See Also

mergePermutationAndObservation for detail description, in the Value section, of the OBSERVATION section of the methylInheritanceAllResults object.

Examples

```
## Load methylation information
data(samplesForTransgenerationalAnalysis)

## Run an observation analysis
runObservation(methylKitData = samplesForTransgenerationalAnalysis,
    outputDir = "test", type = "sites", vSeed = 221)

## Load the results
results <- loadAllRDSResults(analysisResultsDir = "test",
    permutationResultsDir = NULL, doingSites = TRUE, doingTiles = FALSE)

## Print the results
results

## Remove directory
if (dir.exists("test")) {
    unlink("test", recursive = TRUE, force = FALSE)
}
```

---

runOnePermutationOnAllGenerations

Run the analysis on one permutation dataset, including all generations, using methylKit *package*

Description

Run CpG site or region analysis using the methylKit package for each generation present in the dataset. The intersection of conserved elements is obtained for each group of two consecutive generations, as well as, for larger group subset. The output of the analysis is saved in a RDS file when an directory is specified.

Usage

```
runOnePermutationOnAllGenerations(
  id,
  methylInfoForAllGenerations,
  type = c("both", "sites", "tiles"),
```

```
  outputDir = NULL,
  nbrCoresDiffMeth = 1,
  minReads = 10,
  minMethDiff = 10,
  qvalue = 0.01,
  maxPercReads = 99.9,
  destrand = FALSE,
  minCovBasesForTiles = 0,
  tileSize = 1000,
  stepSize = 1000,
  restartCalculation,
  saveInfoByGeneration
)
```

Arguments

id	an integer, the unique identification of the permutation. When id is 0, the analysis is done on the real dataset.
methylInfoForAllGenerations	a list of methylRawList entries. Each methylRawList entry must contain all the methylRaw entries related to one generation (first entry = first generation, second entry = second generation, etc..). The number of generations must correspond to the number of entries in the methylKitData. At least 2 generations must be present to make a permutation analysis. More information can be found in the methylKit package.
type	One of the "sites","tiles" or "both" strings. Specifies the type of differentially methylated elements should be returned. For retrieving differentially methylated bases type="sites"; for differentially methylated regions type="tiles". Default: "both".
outputDir	a string, the name of the directory that will contain the results of the permutation. If the directory does not exist, it will be created.
nbrCoresDiffMeth	a positive integer, the number of cores to use for parallel differential methylation calculations.Parameter used for both sites and tiles analysis. The parameter corresponds to the num.cores parameter in the package methylKit. Default: 1 and always 1 for Windows.
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.
minMethDiff	a positive integer betwwen [0,100], the absolute value of methylation percentage change between cases and controls. The parameter correspond to the difference parameter in the package methylKit. Default: 10.
qvalue	a positive double inferior to 1, the cutoff for qvalue of differential methylation statistic. Default: 0.01.
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. Default: 99.9.

destrand a logical, when TRUE will merge reads on both strands of a CpG dinucleotide to provide better coverage. Only advised when looking at CpG methylation. Parameter used for both sites and tiles analysis. Default: FALSE.

minCovBasesForTiles
a non-negative integer, the minimum number of bases to be covered in a given tiling window. The parameter corresponds to the cov.bases parameter in the package methylKit. Only used when doingTiles = TRUE. Default: 0.

tileSize a positive integer, the size of the tiling window. The parameter corresponds to the win.size parameter in the methylKit package. Only used when doingTiles = TRUE. Default: 1000.

stepSize a positive integer, the step size of tiling windows. The parameter corresponds to the stepSize parameter in the methylKit package. Only used when doingTiles = TRUE. Default: 1000.

restartCalculation
a logical, when TRUE, only permutations that don't have a RDS result final are run.

saveInfoByGeneration
a logical, when TRUE, the information about differentially methylated sites and tiles for each generation is saved in a RDS file. The information is saved in a different file for each permutation. The files are = saved in the outputDir.

Value

a list containing the following elements:

- SITES Only present when type = "sites" or "both", a list containing:
  - i2 a list containing:
    - HYPER a list of integer, the number of conserved hyper differentially methylated sites between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc..
    - HYPO a list of integer, the number of conserved hypo differentially methylated sites between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc..
  - iAll a list containing:
    - HYPER a list of integer, the number of conserved hyper differentially methylated sites between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc..The number of entries depends of the number of generations.
    - HYPO a list of integer, the number of conserved hypo differentially methylated sites between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc..The number of entries depends of the number of generations.

- TILES Only present when type = "tiles" or "both", a list containing: itemize i2 a list containing:
  - HYPER a list of integer, the number of conserved hyper differentially methylated positions between two consecutive generations. The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc..
  - HYPO a list of integer, the number of conserved hypo differentially methylated positions between two consecutive generations.The first element represents the intersection of the first and second generations; the second element, the intersection of the second and third generations; etc..

  iAll a list containing:
  - HYPER a list of integer, the number of conserved hyper differentially methylated positions between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc..The number of entries depends of the number of generations.
  - HYPO a list of integer, the number of conserved hypo differentially methylated positions between three or more consecutive generations. The first element represents the intersection of the first three generations; the second element, the intersection of the first fourth generations; etc..The number of entries depends of the number of generations.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

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

## Run a permutation analysis
methylInheritance:::runOnePermutationOnAllGenerations(id = 2,
    methylInfoForAllGenerations = samplesForTransgenerationalAnalysis,
    type = "tiles", outputDir = NULL,
    nbrCoresDiffMeth = 1, minReads = 10, minMethDiff = 10, qvalue = 0.01,
    maxPercReads = 99.9, destrand = FALSE, minCovBasesForTiles = 0,
    tileSize = 1000, stepSize = 1000, restartCalculation = FALSE)
```

---

runPermutation *Run all permutations on the specified multi-generational dataset*

Description

Run a permutation analysis, based on Monte Carlo sampling, for testing the hypothesis that the number of conserved differentially methylated elements (sites, tiles or both), between several generations, is associated to an effect inherited from a treatment and that stochastic effect can be dismissed.

The multi-generational dataset or the name of the RDS file that contains the dataset can be used as input.

The observation analysis can also be run (optional). All permutation results are saved in RDS files.

Usage

```
runPermutation(
  methylKitData,
  type = c("both", "sites", "tiles"),
  outputDir = "output",
  runObservationAnalysis = TRUE,
  nbrPermutations = 1000,
  nbrCores = 1,
  nbrCoresDiffMeth = 1,
  minReads = 10,
  minMethDiff = 10,
  qvalue = 0.01,
  maxPercReads = 99.9,
  destrand = FALSE,
  minCovBasesForTiles = 0,
  tileSize = 1000,
  stepSize = 1000,
  vSeed = -1,
  restartCalculation = FALSE,
  saveInfoByGeneration = FALSE
)
```

Arguments

`methylKitData`	a `list` of `methylRawList` entries or the name of the RDS file containing the `list`. Each `methylRawList` entry must contain all the `methylRaw` entries related to one generation (first entry = first generation, second entry = second generation, etc..). The number of generations must correspond to the number of entries in the `methylKitData`. At least 2 generations must be present to make a permutation analysis. More information can be found in the methylKit package.
`type`	One of the "sites","tiles" or "both" strings. Specifies the type of differentially methylated elements should be returned. For retrieving differentially methylated bases type="sites"; for differentially methylated regions type="tiles". Default: "both".
`outputDir`	a string, the name of the directory that will contain the results of the permutation. If the directory does not exist, it will be created. Default: `"output"`.
`runObservationAnalysis`	a `logical`, when `runObservationAnalysis = TRUE`, a CpG analysis on the observed dataset is done. Default: `TRUE`.
`nbrPermutations`	a positive `integer`, the total number of permutations that is going to be done. Default: `1000`.

nbrCores
: a positive integer, the number of cores to use when processing the analysis. Default: 1 and always 1 for Windows.

nbrCoresDiffMeth
: a positive integer, the number of cores to use for parallel differential methylation calculations. The parameter is used for both sites and tiles analysis. The parameter corresponds to the num.cores parameter in the package methylKit. Default: 1 and always 1 for Windows.

minReads
: a positive integer Bases and regions having lower coverage than this count are discarded. The parameter corresponds to the lo.count parameter in the package methylKit.

minMethDiff
: a positive double between [0,100], the absolute value of methylation percentage change between cases and controls. The parameter corresponds to the difference parameter in the methylKit package. Default: 10.

qvalue
: a positive double between [0,1], the cutoff for qvalue of differential methylation statistics. Default: 0.01.

maxPercReads
: a double between [0,100], the percentile of read counts that is going to be used as an upper cutoff. Bases or regions having higher coverage than this percentile are discarded. The parameter is used for both CpG sites and tiles analysis. The parameter corresponds to the hi.perc parameter in the package methylKit. Default: 99.9.

destrand
: a logical, when TRUE will merge reads on both strands of a CpG dinucleotide to provide better coverage. Only advised when looking at CpG methylation. The parameter is used for both CpG sites and tiles analysis. Default: FALSE.

minCovBasesForTiles
: a non-negative integer, the minimum number of bases to be covered in a given tiling window. The parameter corresponds to the cov.bases parameter in the package methylKit. Only used when doingTiles = TRUE. Default: 0.

tileSize
: a positive integer, the size of the tiling window. The parameter corresponds to the win.size parameter in the package methylKit. Only used when doingTiles = TRUE. Default: 1000.

stepSize
: a positive integer, the step size of tiling windows. The parameter corresponds to the stepSize parameter in the package methylKit. Only used when doingTiles = TRUE. Default: 1000.

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.

restartCalculation
: a logical, when TRUE, only permutations that don't have an associated RDS result file are run. Useful to restart a permutation analysis that has been interrupted. Beware that the parameters have to be identical except for this one.

saveInfoByGeneration
: a logical, when TRUE, the information about differentially methylated sites and tiles for each generation is saved in a RDS file. The information is saved in a different file for each permutation. The files are saved in the directory specified by the outputDir parameter.

Value

0.

Author(s)

Astrid Deschenes, Pascal Belleau

See Also

mergePermutationAndObservation for detail description, in the Value section, of the methylInheritanceAllResults object as well as its PERMUTATION section.

Examples

```
## Load methylKit information
data(samplesForTransgenerationalAnalysis)

## Run a permutation analysis using the methylKit dataset
## A real analysis would require a much higher number of permutations
runPermutation(methylKitData = samplesForTransgenerationalAnalysis,
    outputDir = "test_01", runObservationAnalysis = FALSE, type = "sites",
    nbrPermutations = 2, vSeed = 221)

## Get results
results_01 <- loadAllRDSResults(analysisResultsDir = NULL,
    permutationResultsDir = "test_01", doingSites = TRUE,
    doingTiles = FALSE)

## Remove results directory
if (dir.exists("test_01")) {
    unlink("test_01", recursive = TRUE, force = TRUE)
}

## Path to a methylKit RDS file
methylFile <- system.file("extdata", "methylObj_001.RDS",
    package = "methylInheritance")

## Run a permutation analysis using RDS file name
## A real analysis would require a much higher number of permutations
runPermutation(methylKitData = methylFile, type = "tiles",
    outputDir = "test_02", nbrPermutations = 2, minCovBasesForTiles = 10,
    vSeed = 2001)

## Get results
results_02 <- loadAllRDSResults(analysisResultsDir = NULL,
    permutationResultsDir = "test_02", doingSites = FALSE,
    doingTiles = TRUE)

## Remove results directory
if (dir.exists("test_02")) {
    unlink("test_02", recursive = TRUE, force = TRUE)
}
```

samplesForTransgenerationalAnalysis
: *All samples information, formated by* `methylKit`*, in a* `methylRawList` *format (for demo purpose).*

Description

The object is a `list` with 3 entries. Each entry corresponds to the information for one generation (first entry = first generation, etc..) stored in a `methylRawList`. There are 12 samples (6 controls and 6 cases) for each generation. Each sample information is stored in a `methylRaw` object.

Usage

```
data(samplesForTransgenerationalAnalysis)
```

Format

A `list` containing three `methylRawList` objects. Each `methylRawList` contains the information for one generation (first entry = first generation, etc..). Each sample information is stored in a `methylRaw` object. There is `methylRaw` objects (6 controls and 6 cases) in each generation.

Details

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

Value

A `list` containing three `methylRawList` objects. Each `methylRawList` contains the information for one generation (first entry = first generation, etc..). Each sample information is stored in a `methylRaw` object. There is `methylRaw` objects (6 controls and 6 cases) in each generation.

See Also

- `runPermutation` for running a permutation analysis, and optionally an observation analysis, using multi-generational dataset

Examples

```
## Loading dataset
data(samplesForTransgenerationalAnalysis)

## Run a permutation analysis
runPermutation(methylKitData = samplesForTransgenerationalAnalysis,
    type = "tiles", nbrPermutations = 2, vSeed = 2332)
```

`saveInterGenerationResults` *Save the result of on CpG site or tile analysis on all generations. The anaysis can come from observed or shuffled dataset. Each case is saved with a different extension.*

Description

Save the result of on CpG site or tile analysis on all generations. The results are saved in a RDS file. The anaysis can have been done on the observed or shuffled dataset. Each permutation is saved using its identifiant in the file name.

Usage

```
saveInterGenerationResults(
  outputDir,
  permutationID,
  type = c("sites", "tiles"),
  interGenerationResult
)
```

Arguments

`outputDir`	a string of `character`, the name of the directory that will contain the results of the permutation. The name should end with a slash. The directory should already exists.
`permutationID`	an `integer`, the identifiant of the permutation. When the `permutationID = 0`, the results are considered as the observed results and are saved in a file with the "_observed_results.RDS" extension. When the `permutationID != 0`, the results are considered as permutation results and are saved in a file with the "_permutation_permutationID.RDS" extension.
`type`	One of the `"sites"` or `"tiles"` strings. Specifies the type of differentially methylated elements should be saved. Default: `"sites"`.
`interGenerationResult`	a `list` that corresponds to the output of the `interGeneration` function, the result of on CpG site or tile analysis on all generations.

Value

`0` indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Load permutation results on sites

permutationResultsFile <- system.file("extdata",
    "permutationResultsForSites.RDS", package="methylInheritance")
permutationResults <- readRDS(permutationResultsFile)

## Transform result to GRanges
resultsGR <- methylInheritance:::getGRangesFromMethylDiff(methDiff =
    permutationResults, pDiff = 10, qvalue = 0.01, type = "hyper")

## Extract inter-generationally conserved sites
interGenerationResult <- methylInheritance:::interGeneration(resultsGR)

## Create directories
dir.create("TEST", showWarnings = TRUE)
dir.create("TEST/SITES", showWarnings = TRUE)

## Save results
methylInheritance:::saveInterGenerationResults(
    outputDir = "TEST/", permutationID=100, type = "sites",
    interGenerationResult = interGenerationResult)
```

---

validateExtractInfo *Validation of some parameters of the* extractInfo *function*

Description

Validation of some parameters needed by the public extractInfo function.

Usage

```
validateExtractInfo(allResults, type, inter, position)
```

Arguments

allResults	a list as created by the runPermutation or the loadAllRDSResults functions.
type	One of the "sites" or "tiles" strings. Specifies the type of differentially methylated elements should be returned. For retrieving differentially methylated bases type = "sites"; for differentially methylated regions type = "tiles".
inter	One of the "i2" or "iAll" strings. Specifies the type of intersection should be returned. For retrieving intersection results between two consecutive generations inter = "i2"; for intersection results between three generations or more inter = "iAll".
position	a positive integer, the position in the list where the information will be extracted. The position must be an existing position inside allResults

Value

`0` indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes

Examples

```
## Load dataset
data(methylInheritanceResults)

## The function returns 0 when all paramaters are valid
methylInheritance:::validateExtractInfo(
    allResults = methylInheritanceResults, type = "sites",
    inter = "i2", 2)

## The function raises an error when at least one paramater is not valid
## Not run: methylInheritance:::validateExtractInfo(
    allResults = methylInheritanceResults, type = "sites",
    inter = "i2", 12)
## End(Not run)
```

`validateLoadConvergenceData` *Validation of some parameters of the* `loadConvergenceData` *function*

Description

Validation of some parameters needed by the public `loadConvergenceData` function.

Usage

```
validateLoadConvergenceData(
  analysisResultsDir,
  permutationResultsDir,
  position,
  by
)
```

Arguments

`analysisResultsDir`
: a `character` string, the path to the directory that contains the analysis results. The path can be the same as for the `permutatioNResultsDir` parameter.

`permutationResultsDir`
: a `character` string, the path to the directory that contains the permutation results. The path can be the same as for the `analysisResultsDir` parameter.

position — a positive integer, the position in the list where the information will be extracted.

by — a integer, the increment of the number of permutations where the significant level is tested. Default: 100.

Value

0 indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Get the name of the directory where files are stored
filesDir <- system.file("extdata", "TEST", package="methylInheritance")

## Merge permutation and observation results
methylInheritance:::validateLoadConvergenceData(analysisResultsDir =
    filesDir, permutationResults = filesDir, position = 1, by = 1)

## The function raises an error when at least one paramater is not valid
## Not run: methylInheritance:::validateLoadConvergenceData(
    analysisResultsDir = filesDir, permutationResults = filesDir,
    position = "hello", by = 1))
## End(Not run)
```

---

validateMergePermutationAndObservation *Validation of some parameters of the* mergePermutationAndObservation *function*

Description

Validation of some parameters needed by the public mergePermutationAndObservation function.

Usage

```
validateMergePermutationAndObservation(permutationResults, observationResults)
```

Arguments

permutationResults — a list with 1 entry called PERMUTATION. The PERMUTATION entry is a list with a number of entries corresponding to the number of permutations that have been processed. Each entry contains the result of one permutation.

observationResults
: a list with 1 entry called OBSERVATION. The OBSERVATION entry is a list containing the result obtained with the observed dataset (not shuffled).

Value

0 indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes

Examples

```
## Create a observation result
observed <- list()
observed[["OBSERVATION"]] <- list()
observed[["OBSERVATION"]][["SITES"]] <- list()
observed[["OBSERVATION"]][["SITES"]][["i2"]] <- list(HYPER = list(11, 10),
    HYPO = list(13, 12))
observed[["OBSERVATION"]][["SITES"]][["iAll"]] <- list(HYPER = list(1),
    HYPO = list(3))

## Create a permutation result containing only 1 permutation result
## Real perumtations results would have more entries
permutated <- list()
permutated[["PERMUTATION"]] <- list()
permutated[["PERMUTATION"]][[1]] <- list()
permutated[["PERMUTATION"]][[1]][["SITES"]] <- list()
permutated[["PERMUTATION"]][[1]][["SITES"]][["i2"]] <- list(HYPER =
    list(11, 12), HYPO = list(8, 11))
permutated[["PERMUTATION"]][[1]][["SITES"]][["iAll"]] <- list(HYPER =
    list(0), HYPO = list(1))

## Merge permutation and observation results
methylInheritance:::validateMergePermutationAndObservation(
    permutationResults = permutated, observationResults = observed)

## The function raises an error when at least one paramater is not valid
## Not run: methylInheritance:::validateMergePermutationAndObservation(
    permutationResults = permutated, observationResults = NULL)
## End(Not run)
```

---

validateRunObservation

Validation of some parameters of the runObservation function

Description

Validation of some parameters needed by the public runObservation function.

Usage

```
validateRunObservation(
  methylKitData,
  type,
  outputDir,
  nbrCoresDiffMeth,
  minReads,
  minMethDiff,
  qvalue,
  maxPercReads,
  destrand,
  minCovBasesForTiles,
  tileSize,
  stepSize,
  vSeed,
  restartCalculation,
  saveInfoByGeneration
)
```

Arguments

methylKitData	a list of methylRawList entries or the name of the RDS file containing the list. Each methylRawList contains all the methylRaw entries related to one generation (first entry = first generation, second entry = second generation, etc..). The number of generations must correspond to the number of entries in the methylKitData. At least 2 generations must be present to calculate the conserved elements. More information can be found in the methylKit package.
type	One of the "sites","tiles" or "both" strings. Specifies the type of differentially methylated elements should be returned. For retrieving differentially methylated bases type="sites"; for differentially methylated regions type="tiles". Default: "both".
outputDir	a string, the name of the directory that will contain the results of the permutation. If the directory does not exist, it will be created.
nbrCoresDiffMeth	a positive integer, the number of cores to use for parallel differential methylation calculations.Parameter used for both sites and tiles analysis. The parameter corresponds to the num.cores parameter in the methylKit package.
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.
minMethDiff	a positive double betwwen [0,100], the absolute value of methylation percentage change between cases and controls. The parameter correspond to the difference parameter in the methylKit package.
qvalue	a positive double betwwen [0,1], the cutoff for qvalue of differential methylation statistic.

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.

destrand
: a logical, when TRUE will merge reads on both strands of a CpG dinucleotide to provide better coverage. Only advised when looking at CpG methylation. Parameter used for both CpG sites and tiles analysis.

minCovBasesForTiles
: a non-negative integer, the minimum number of bases to be covered in a given tiling window. The parameter corresponds to the cov.bases parameter in the package methylKit. Only used when doingTiles = TRUE. Default: 0.

tileSize
: a positive integer, the size of the tiling window. The parameter corresponds to the win.size parameter in the methylKit package. Only used when doingTiles = TRUE.

stepSize
: a positive integer, the step size of tiling windows. The parameter corresponds to the stepSize parameter in the methylKit package. Only used when doingTiles = TRUE.

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.

restartCalculation
: a logical, when TRUE, only permutations that don't have an associated RDS result file are run. Useful to restart a permutation analysis that has been interrupted.

saveInfoByGeneration
: a logical, when TRUE, the information about differentially methylated sites and tiles for each generation is saved in a RDS file. The information is saved in a different file for each permutation. The files are only saved when the outputDir is not NULL.

Value

0 indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes

Examples

```
## Load dataset
data(samplesForTransgenerationalAnalysis)

## The function returns 0 when all paramaters are valid
methylInheritance:::validateRunObservation(
    methylKitData = samplesForTransgenerationalAnalysis, type = "sites",
    outputDir = "test", nbrCoresDiffMeth = 1, minReads = 10,
    minMethDiff = 25, qvalue = 0.01,
    maxPercReads = 99.9, destrand = TRUE, minCovBasesForTiles = 10,
```

```
    tileSize = 1000, stepSize = 500, vSeed = 12, restartCalculation = TRUE,
    saveInfoByGeneration = FALSE)

## The function raises an error when at least one paramater is not valid
## Not run: methylInheritance:::validateRunObservation(
    methylKitData = samplesForTransgenerationalAnalysis,
    type = "tiles", outputDir = "test_02", nbrCoresDiffMeth = 1,
    minReads = "HI", minMethDiff = 25, qvalue = 0.01,
    maxPercReads = 99.9, destrand = TRUE, minCovBasesForTiles = 10,
    tileSize = 1000, stepSize = 500, vSeed = 12, restartCalculation = FALSE,
    saveInfoByGeneration = FALSE)
## End(Not run)
```

---

validateRunPermutation *Parameters validation for the* runPermutation *function*

Description

Validation of all parameters needed by the public runPermutation function.

Usage

```
validateRunPermutation(
  methylKitData,
  type,
  outputDir,
  runObservedAnalysis,
  nbrPermutations,
  nbrCores,
  nbrCoresDiffMeth,
  minReads,
  minMethDiff,
  qvalue,
  maxPercReads,
  destrand,
  minCovBasesForTiles,
  tileSize,
  stepSize,
  vSeed,
  restartCalculation,
  saveInfoByGeneration
)
```

Arguments

methylKitData	a list of methylRawList entries or the name of the RDS file containing the list. Each methylRawList entry must contain all the methylRaw entries related to one generation (first entry = first generation, second entry = second generation, etc..). The number of generations must correspond to the number of entries in the methylKitData. At least 2 generations must be present to do a permutation analysis. More information can be found in the methylKit package.
type	One of the "sites","tiles" or "both" strings. Specifies the type of differentially methylated elements should be returned. For retrieving differentially methylated bases type="sites"; for differentially methylated regions type="tiles". Default: "both".
outputDir	a string, the name of the directory that will contain the results of the permutation. If the directory does not exist, it will be created.
runObservedAnalysis	a logical, when runObservedAnalysis = TRUE, a CpG analysis on the observed dataset is done.
nbrPermutations	a positive integer, the total number of permutations that is going to be done.
nbrCores	a positive integer, the number of cores to use when processing the analysis.
nbrCoresDiffMeth	a positive integer, the number of cores to use for parallel differential methylation calculations.Parameter used for both sites and tiles analysis. The parameter corresponds to the num.cores parameter in the methylKit package.
minReads	a positive integer Bases and regions having lower coverage than this count are discarded. The parameter corresponds to the lo.count parameter in the methylKit package.
minMethDiff	a positive double betwwen [0,100], the absolute value of methylation percentage change between cases and controls. The parameter corresponds to the difference parameter in the methylKit package.
qvalue	a positive double betwwen [0,1], the cutoff for qvalue of differential methylation statistic. TODO
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.
destrand	a logical, when TRUE will merge reads on both strands of a CpG dinucleotide to provide better coverage. Only advised when looking at CpG methylation. Parameter used for both CpG sites and tiles analysis.
minCovBasesForTiles	a non-negative integer, the minimum number of bases to be covered in a given tiling window. The parameter corresponds to the cov.bases parameter in the package methylKit. Only used when doingTiles = TRUE. Default: 0.
tileSize	a positive integer, the size of the tiling window. The parameter corresponds to the win.size parameter in the methylKit package. Only used when doingTiles = TRUE.

stepSize — a positive integer, the step size of tiling windows. The parameter corresponds to the stepSize parameter in the methylKit package. Only used when doingTiles = TRUE.

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.

restartCalculation — a logical, when TRUE, only permutations that don't have an associated RDS result file are run. Useful to restart a permutation analysis that has been interrupted.

saveInfoByGeneration — a logical, when TRUE, the information about differentially methylated sites and tiles for each generation is saved in a RDS file. The information is saved in a different file for each permutation. The files are only saved when the outputDir is not NULL.

Value

0 indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes

Examples

```
## Load dataset
data(samplesForTransgenerationalAnalysis)

## The function returns 0 when all paramaters are valid
methylInheritance:::validateRunPermutation(
    methylKitData = samplesForTransgenerationalAnalysis, type = "sites",
    outputDir = "test", runObservedAnalysis = TRUE,
    nbrPermutations = 10000, nbrCores = 1,
    nbrCoresDiffMeth = 1, minReads = 10, minMethDiff = 25, qvalue = 0.01,
    maxPercReads = 99.9, destrand = TRUE, minCovBasesForTiles = 10,
    tileSize = 1000, stepSize = 500, vSeed = 12, restartCalculation = FALSE,
    saveInfoByGeneration = FALSE)

## The function raises an error when at least one paramater is not valid
## Not run: methylInheritance:::validateRunPermutation(
    methylKitData = "HI", type = "tiles", outputDir = "test",
    runObservedAnalysis = FALSE, nbrPermutations = 10000, nbrCores = 1,
    nbrCoresDiffMeth = 1, minReads = 10, minMethDiff = 25, qvalue = 0.01,
    maxPercReads = 99.9, destrand = TRUE, minCovBasesForTiles = 10,
    tileSize = 1000, stepSize = 500, vSeed = 12, restartCalculation = FALSE,
    saveInfoByGeneration = FALSE)
## End(Not run)
```

Index