Package: SOMNiBUS (via r-universe)

September 13, 2024

Title Smooth modeling of bisulfite sequencing

Version 1.13.0

Description This package aims to analyse count-based methylation data on predefined genomic regions, such as those obtained by targeted sequencing, and thus to identify differentially methylated regions (DMRs) that are associated with phenotypes or traits. The method is built a rich flexible model that allows for the effects, on the methylation levels, of multiple covariates to vary smoothly along genomic regions. At the same time, this method also allows for sequencing errors and can adjust for variability in cell type mixture.

License MIT + file LICENSE

URL https://github.com/kaiqiong/SOMNiBUS

BugReports https://github.com/kaiqiong/SOMNiBUS/issues

Depends R (>= 4.1.0)

Imports Matrix, mgcv, stats, VGAM, IRanges, GenomeInfoDb, GenomicRanges, rtracklayer, S4Vectors, BiocManager, annotatr, yaml, utils, bsseq, reshape2, data.table, ggplot2, tidyr,

Suggests BiocStyle, covr, devtools, dplyr, knitr, magick, rmarkdown, testthat, TxDb.Hsapiens.UCSC.hg38.knownGene, TxDb.Hsapiens.UCSC.hg19.knownGene, org.Hs.eg.db,

VignetteBuilder knitr

biocViews DNAMethylation, Regression, Epigenetics, DifferentialMethylation, Sequencing, FunctionalPrediction

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Repository https://bioc.r-universe.dev

RemoteUrl https://github.com/bioc/SOMNiBUS

RemoteRef HEAD

RemoteSha ad81567efc246148e820bfb6bcdf313574a5fb29

Contents

`binomRegMethModel` *A smoothed-EM algorithm to estimate covariate effects and test regional association in Bisulfite Sequencing-derived methylation data*

Description

This function fits a (dispersion-adjusted) binomial regression model to regional methylation data, and reports the estimated smooth covariate effects and regional p-values for the test of DMRs (differentially methylation regions). Over or under dispersion across loci is accounted for in the model by the combination of a multiplicative dispersion parameter (or scale parameter) and a sample-specific random effect.

This method can deal with outcomes, i.e. the number of methylated reads in a region, that are contaminated by known false methylation calling rate (`p0`) and false non-methylation calling rate (`1-p1`).

The covariate effects are assumed to smoothly vary across genomic regions. In order to estimate them, the algorithm first represents the functional parameters by a linear combination of a set of restricted cubic splines (with dimension `n.k`), and a smoothness penalization term which depends on the smoothing parameters `lambdas` is also added to control smoothness. The estimation is performed by an iterated EM algorithm. Each M step constitutes an outer Newton's iteration to estimate smoothing parameters `lambdas` and an inner P-IRLS iteration to estimate spline coefficients `alpha` for the covariate effects. Currently, the computation in the M step depends the implementation of `gam()` in package `mgcv`.

Usage

```
binomRegMethModel(
  data,
  n.k,
  p0 = 0.003,
  p1 = 0.9,
  Quasi = TRUE,
  epsilon = 10^(-6),
  epsilon.lambda = 10^(-3),
  maxStep = 200,
  binom.link = "logit",
  method = "REML",
  covs = NULL,
  RanEff = TRUE,
  reml.scale = FALSE,
  scale = -2,
  verbose = TRUE
)
```

Arguments

`data`	a data frame with rows as individual CpGs appearing in all the samples. The first 4 columns should contain the information of `Meth_Counts` (methylated counts), `Total_Counts` (read depths), `Position` (Genomic position for the CpG site) and `ID` (sample ID). The covariate information, such as disease status or cell type composition, are listed in column 5 and onwards.
`n.k`	a vector of basis dimensions for the intercept and individual covariates. `n.k` specifies an upper limit of the degrees of each functional parameters. The length of n.k should equal to the number of covariates plus 1 (for the intercept)). We recommend basis dimensions n.k, approximately equal to the number of unique CpGs in the region divided by 20. This parameter will be computed automatically, when several regions are generated by the partitioning function.
`p0`	the probability of observing a methylated read when the underlying true status is unmethylated. `p0` is the rate of false methylation calls, i.e. false positive rate.
`p1`	the probability of observing a methylated read when the underlying true status is methylated. `1-p1` is the rate of false non-methylation calls, i.e. false negative rate.
`Quasi`	whether a Quasi-likelihood estimation approach will be used; in other words, whether a multiplicative dispersion is added in the model or not.
`epsilon`	numeric; stopping criterion for the closeness of estimates of spline coefficients from two consecutive iterations.
`epsilon.lambda`	numeric; stopping criterion for the closeness of estimates of smoothing parameter `lambda` from two consecutive iterations.
`maxStep`	the algorithm will step if the iteration steps exceed `maxStep`.
`binom.link`	the link function used in the binomial regression model; the default is the logit link.

method	the method used to estimate the smoothing parameters. The default is the 'REML' method which is generally better than prediction based criterion GCV.cp.
covs	a vector of covariate names. The covariates with names in covs will be included in the model and their covariate effects will be estimated. The default is to fit all covariates in dat
RanEff	whether sample-level random effects are added or not
reml.scale	whether a REML-based scale (dispersion) estimator is used. The default is Fletcher-based estimator.
scale	negative values mean scale parameter should be estimated; if a positive value is provided, a fixed scale will be used.
verbose	logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

This function return a list including objects:

- est: estimates of the spline basis coefficients alpha
- lambda: estimates of the smoothing parameters for each functional parameters
- est.pi: predicted methylation levels for each row in the input data
- ite.points: estimates of est, lambda at each EM iteration
- cov1: estimated variance-covariance matrix of the basis coefficients alphas
- reg.out: regional testing output obtained using Fletcher-based dispersion estimate; an additional 'ID' row would appear if RanEff is TRUE
- reg.out.reml.scale:regional testing output obtained using REML-based dispersion estimate;
- reg.out.gam:regional testing output obtained using (Fletcher-based) dispersion estimate from mgcv package;
- phi_fletcher: Fletcher-based estimate of the (multiplicative) dispersion parameter;
- phi_reml: REML-based estimate of the (multiplicative) dispersion parameter;
- phi_gam: Estimated dispersion parameter reported by mgcv;
- SE.out: a matrix of the estimated pointwise Standard Errors (SE); number of rows are the number of unique CpG sites in the input data and the number of columns equal to the total number of covariates fitted in the model (the first one is the intercept);
- SE.out.REML.scale: a matrix of the estimated pointwise Standard Errors (SE); the SE calculated from the REML-based dispersion estimates
- uni.pos: the genomic postions for each row of CpG sites in the matrix SE.out;
- Beta.out: a matrix of the estimated covariate effects beta(t), where t denotes the genomic positions;
- ncovs: number of functional paramters in the model (including the intercept);
- sigma00: estimated variance for the random effect if RanEff is TRUE; NA if RanEff is FALSE.

Author(s)

Kaiqiong Zhao

See Also

gam

Examples

```
#------------------------------------------------------------#
data(RAdat)
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
out <- binomRegMethModel(
  data=RAdat.f, n.k=rep(5,3), p0=0.003307034, p1=0.9,
  epsilon=10^(-6), epsilon.lambda=10^(-3), maxStep=200
)
```

---

binomRegMethModelPlot *Plot the smooth covariate effect*

Description

This function accepts an output object from function binomRegMethModel and print out a plot of the estimated effect across the region for each test covariate.

Usage

```
binomRegMethModelPlot(
  BEM.obj,
  mfrow = NULL,
  same.range = FALSE,
  title = "Smooth covariate effects",
  covs = NULL,
  save = NULL,
  verbose = TRUE
)
```

Arguments

BEM.obj	an output object from function binomRegMethModel
mfrow	A vector of the form c(nr, nc). Subsequent figures will be drawn in an nr-by-nc array on the device.
same.range	specify whether the plots should be in the same vertical scale
title	the text for the title
covs	a vector of covariate names. The covariates with names in covs will be included in the plot. When the value is set to NULL all the covariates and the Intercept will be represented. The default value is NULL.

`save`	file name to create on disk. When the value is set to NULL, the plot is not saved. The default value is NULL.
`verbose`	logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

This function prints out a plot of smooth covariate effects and its pointwise confidence intervals

Author(s)

Kaiqiong Zhao, Audrey Lemaçon

Examples

```
#------------------------------------------------------------#
data(RAdat)
head(RAdat)
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
out <- binomRegMethModel(
  data=RAdat.f, n.k=rep(5, 3), p0=0.003307034, p1=0.9,
  epsilon=10^(-6), epsilon.lambda=10^(-3), maxStep=200,
  Quasi = FALSE, RanEff = FALSE
)
binomRegMethModelPlot(out, same.range=FALSE)
```

`binomRegMethModelPred` *A smoothed-EM algorithm to estimate covariate effects and test regional association in Bisulfite Sequencing-derived methylation data*

Description

This function returns the predicted methylation levels

Usage

```
binomRegMethModelPred(
  BEM.obj,
  newdata = NULL,
  type = "proportion",
  verbose = TRUE
)
```

Arguments

`BEM.obj`	an output from the function `binomRegMethModel`
`newdata`	the data set whose predictions are calculated; with columns 'Position', covariate names matching the BEM.obj
`type`	return the predicted methylation proportion or the predicted response (in logit or other binom.link scale)
`verbose`	logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

This function returns the predicted methylation levels

Author(s)

Kaiqiong Zhao

Examples

```
#------------------------------------------------------------#
data(RAdat)
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
out <- binomRegMethModel(
  data=RAdat.f, n.k=rep(5, 3), p0=0.003307034, p1=0.9,
  epsilon=10^(-6), epsilon.lambda=10^(-3), maxStep=200,
  Quasi = FALSE, RanEff = FALSE
)
binomRegMethModelPred(out)
```

`binomRegMethModelSim` *Simulate Bisulfite sequencing data from specified smooth covariate effects*

Description

Simulate Bisulfite sequencing data from a Generalized Additive Model with functional parameters varying with the genomic position. Both the true methylated counts and observed methylated counts are generated, given the error/conversion rate parameters `p0` and `p1`. In addition, the true methylated counts can be simulated from a binomial or a dispersed binomial distribution (Beta-binomial distribution).

Usage

```
binomRegMethModelSim(
  n,
  posit,
  theta.0,
  beta,
  phi,
  random.eff = FALSE,
  mu.e = 0,
  sigma.ee = 1,
  p0 = 0.003,
  p1 = 0.9,
  X,
  Z,
  binom.link = "logit",
  verbose = TRUE
)
```

Arguments

`n`	sample size
`posit`	a numeric vector of size `p` (the number of CpG sites in the considered region) containing the genomic positions;
`theta.0`	numeric vector of size `p` which is a functional parameter for the intercept of the GAMM model.
`beta`	numeric vector of size `p` which is a functional parameter for the slope of cell type composition.
`phi`	a vector of length `p` determining the multiplicative dispersion parameter for each loci in a region. The dispersed-Binomial counts are simulated from beta-binomial distribution, so each element of phi has to be greater than 1.
`random.eff`	indicates whether adding the subject-specific random effect term `e`.
`mu.e`	number, the mean of the random effect.
`sigma.ee`	positive number, variance of the random effect
`p0`	the probability of observing a methylated read when the underlying true status is unmethylated. `p0` is the rate of false methylation calls, i.e. false positive rate.
`p1`	the probability of observing a methylated read when the underlying true status is methylated. `1-p1` is the rate of false non-methylation calls, i.e. false negative rate.
`X`	the matrix of the read coverage for each CpG in each sample; a matrix of `n` rows and `p` columns.
`Z`	numeric matrix with `p` columns and `n` rows storing the covariate information.
`binom.link`	the link function used for simulation
`verbose`	logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

The function returns a list of following objects

- S a numeric matrix of n rows and p columns containing the true methylation counts;
- Y a numeric matrix of n rows and p columns containing the observed methylation counts;
- theta a numeric matrix of n rows and p columns containing the methylation parameter (after the logit transformation);
- pi a numeric matrix of n rows and p columns containing the true methylation proportions used to simulate the data.

Author(s)

Kaiqiong Zhao

Examples

```
#------------------------------------------------------------#
data(RAdat)
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
out <- binomRegMethModel(
   data=RAdat.f, n.k=rep(5, 3), p0=0, p1=1,
   epsilon=10^(-6), epsilon.lambda=10^(-3), maxStep=200, RanEff = FALSE
)
Z = as.matrix(RAdat.f[match(unique(RAdat.f$ID), RAdat.f$ID),
c('T_cell', 'RA')])
set.seed(123)
X = matrix(sample(80, nrow(Z)*length(out$uni.pos), replace = TRUE),
nrow = nrow(Z), ncol = length(out$uni.pos))+10
simdat = binomRegMethModelSim(n=nrow(Z), posit= out$uni.pos,
theta.0=out$Beta.out[,1], beta= out$Beta.out[,-1], random.eff=FALSE,
mu.e=0,sigma.ee=1, p0=0.003, p1=0.9,X=X , Z=Z, binom.link='logit',
phi = rep(1, length(out$uni.pos)))
```

---

binomRegMethPredPlot *Plot the predicted methylation levels*

Description

This function accepts the data.frame used as an input for the function binomRegMethModelPred with additional columns containing the predictions generated by the function binomRegMethModelPred and columns containing the name of each experimental group and returns a plot representing the predicted methylation levels according to each experimental group.

Usage

```
binomRegMethPredPlot(
  pred,
  pred.type = "proportion",
  pred.col = "pred",
  group.col = NULL,
  title = "Predicted methylation levels",
  style = NULL,
  save = NULL,
  verbose = TRUE
)
```

Arguments

`pred` — `data.frame` used as an input for the function `binomRegMethModelPred` (with columns 'Position', covariate names matching the original output from the function `binomRegMethModel`) with additional columns containing the predictions generated by the function `binomRegMethModelPred` and columns containing the name of each experimental group. Rows without a valid group name (empty character `""` or `NA`) are ignored

`pred.type` — type of prediction returned by the function `binomRegMethModelPred`: `proportion` or `link.scale`. The default value is "proportion".

`pred.col` — character defines the name of the column containing the prediction values. The default value is "pred".

`group.col` — character defines the name of the column containing the experimental groups. If the group.col is set to NULL, the resulting plot will be a simple scatter plot representing all predicted values disregarding any experimental design. The default value is NULL.

`title` — the text for the title

`style` — named list containing the wanted style (color and line type) for each experimental groups. The first level list is named according each experimental group and for each experimental group there is a list containing the `color` and the `type` of the line. The line types should be among the following types:

- `twodash`,
- `solid`,
- `longdash`,
- `dotted`,
- `dotdash`,
- `dashed`,
- `blank`.

The function accepts color name and its hexadecimal code. The default value is NULL meaning that the colors will be chosen randomly and the line style will be set to `solid`.

`save` — file name to create on disk. When the value is set to NULL, the plot is not saved. The default value is NULL.

`verbose` logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

This function prints out a plot of the predicted methylation levels according to preset experimental groups.

Author(s)

Audrey Lemaçon

Examples

```
#------------------------------------------------------------#
data(RAdat)
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
BEM.obj <- binomRegMethModel(
  data=RAdat.f, n.k=rep(5, 3), p0=0.003307034, p1=0.9,
  epsilon=10^(-6), epsilon.lambda=10^(-3), maxStep=200,
  Quasi = FALSE, RanEff = FALSE, verbose = FALSE
)
pos <- BEM.obj$uni.pos
newdata <- expand.grid(pos, c(0, 1), c(0, 1))
colnames(newdata) <- c("Position", "T_cell", "RA")
my.pred <- binomRegMethModelPred(BEM.obj, newdata, type = "link.scale",
verbose = FALSE)
newdata$group <- ""
newdata[(newdata$RA == 0 & newdata$T_cell == 0),]$group <- "CTRL MONO"
newdata[(newdata$RA == 0 & newdata$T_cell == 1),]$group <- "CTRL TCELL"
newdata[(newdata$RA == 1 & newdata$T_cell == 0),]$group <- "RA MONO"
newdata[(newdata$RA == 1 & newdata$T_cell == 1),]$group <- "RA TCELL"
pred <- cbind(newdata, Pred = my.pred)
style <- list("CTRL MONO" = list(color = "blue", type = "dashed"),
"CTRL TCELL" = list(color = "green", type = "dashed"),
"RA MONO" = list(color = "blue", type = "solid"),
"RA TCELL" = list(color = "green", type = "solid"))
g <- binomRegMethPredPlot(pred, pred.col = "Pred", group.col = "group",
style = style, save = NULL, verbose = FALSE)
```

`formatFromBismark` *Parsing output from the Bismark alignment suite*

Description

This function reads and converts Bismark's 'genome wide cytosine report' and 'coverage' into a `list` of `data.frames` (one per chromosome) to a format compatible with SOMNiBUS' main functions `runSOMNiBUS` and `binomRegMethModel`.

Usage

```
formatFromBismark(..., verbose = TRUE)
```

Arguments

`...`	parameters from `bsseq::read.bismark()` function
`verbose`	logical indicates the level of information provided by the algorithm during the process. The default value is TRUE.

Value

This function returns a `list` of `data.frames` (one per chromosome). Each `data.frame` contains rows as individual CpGs appearing in all the samples. The first 4 columns contain the information of `Meth_Counts` (methylated counts), `Total_Counts` (read depths), `Position` (Genomic position for the CpG site) and `ID` (sample ID). The additional information (such as disease status, sex, age) extracted from the BSseq object are listed in column 5 and onwards and will be considered as covariate information by SOMNiBUS algorithms.

Author(s)

Audrey Lemaçon

See Also

read.bismark for parsing output from the Bismark alignment suite.

Other Parsing functions: `formatFromBSmooth()`, `formatFromBSseq()`

Examples

```
infile <- system.file("extdata/test_data.fastq_bismark.bismark.cov.gz",
package = "bsseq")
dat <- formatFromBismark(infile, verbose = FALSE)
```

`formatFromBSmooth` *Parsing output from the BSmooth alignment suite*

Description

This function reads and converts BSmooth's output into a `list` of `data.frames` (one per chromosome) to a format compatible with SOMNiBUS' main functions `runSOMNiBUS` and `binomRegMethModel`.

Usage

```
formatFromBSmooth(..., verbose = TRUE)
```

Arguments

...	parameters from bsseq::read.bsmooth() function
verbose	logical indicates the level of information provided by the algorithm during the process. The default value is TRUE.

Value

This function returns a list of data.frames (one per chromosome). Each data.frame contains rows as individual CpGs appearing in all the samples. The first 4 columns contain the information of Meth_Counts (methylated counts), Total_Counts (read depths), Position (Genomic position for the CpG site) and ID (sample ID). The additional information (such as disease status, sex, age) extracted from the BSseq object are listed in column 5 and onwards and will be considered as covariate information by SOMNiBUS algorithms.

Author(s)

Audrey Lemaçon

See Also

read.bismark for parsing output from the Bismark alignment suite.

Other Parsing functions: formatFromBSseq(), formatFromBismark()

Examples

```
indir <- system.file("extdata/ev_bt2_tab", package = "SOMNiBUS")
dat <- formatFromBSmooth(indir, verbose = FALSE)
```

---

formatFromBSseq — *Parsing output from the BSseq package*

Description

This function reads and converts a BSseq object into a list of data.frames (one per chromosome) to a format compatible with SOMNiBUS' main functions runSOMNiBUS and binomRegMethModel.

Usage

```
formatFromBSseq(bsseq_dat, verbose = TRUE)
```

Arguments

bsseq_dat	an object of class BSseq.
verbose	logical indicates the level of information provided by the algorithm during the process. The default value is TRUE.

Value

This function returns a list of data.frames (one per chromosome). Each data.frame contains rows as individual CpGs appearing in all the samples. The first 4 columns contain the information of Meth_Counts (methylated counts), Total_Counts (read depths), Position (Genomic position for the CpG site) and ID (sample ID). The additional information (such as disease status, sex, age) extracted from the BSseq object are listed in column 5 and onwards and will be considered as covariate information by SOMNiBUS algorithms.

Author(s)

Audrey Lemaçon

See Also

BSseq for the BSseq class.

Other Parsing functions: formatFromBSmooth(), formatFromBismark()

Examples

```
M <- matrix(1:9, 3,3)
colnames(M) <- c("A1", "A2", "A3")
BStest <- bsseq::BSseq(pos = 1:3, chr = c("chr1", "chr2", "chr1"),
M = M, Cov = M + 2)
dat <- formatFromBSseq(BStest, verbose = FALSE)
```

RAdat *Methylation data from a rheumatoid arthritis study*

Description

A dataset containing methylation levels on one targeted region on chromosome 4 near gene BANK1 from cases with rheumatoid arthritis (RA) and controls.

Usage

```
RAdat
```

Format

A data frame of 5289 rows and 6 columns. Each row represents a CpG site for a sample. Columns include in order:

Meth_Counts Number of methylated reads

Total_Counts Total number of reads; read-depth

Position Genomic position (in bp) for the CpG site

ID indicates which sample the CpG site belongs to

T_cell whether a sample is from T cell or monocyte

RA whether a sample is an RA patient or control

Details

This example data include methylation levels of cell type separated blood samples of 22 rheumatoid arthritis (RA) patients and 21 healthy individuals. In the data set, 123 CpG sites are measured and there are 25 samples from circulating T cells and 18 samples from monocytes.

Source

Dr. Marie Hudson (McGill University)

`RAdat2` *A simulated methylation dataset based on a real data.*

Description

This example data include methylation levels on a region with 208 CpGs for 116 blood samples.

Usage

```
RAdat2
```

Format

A data frame of 6064 rows and 13 columns. Each row represents a CpG site for a sample. Columns include in order:

Meth_Counts Number of methylated reads

Total_Counts Total number of reads; read-depth

Position Genomic position (in bp) for the CpG site

ID indicates which sample the CpG site belongs to

ACPA4 binary indicator for a biomarker anti-citrullinated protein antibody

Age Age

Sex 2-female; 1-male

Smoking 1-current or ex-smoker; 0-non-smoker

Smoking_NA 1-Smoking info is NA; 0-Smoking info is available

PC1 PC1 for the cell type proportions

PC2 PC2 for the cell type proportions

PC3 PC3 for the cell type proportions

PC4 PC4 for the cell type proportions

Source

simulation is based a real data set provided by PI Dr. Sasha Bernatsky (McGill University) runSOMNiBUS *Wrapper function running the smoothed-EM algorithm to estimate covariate effects and test regional association in Bisulfite Sequencing-derived methylation data*

Description

This function splits the methylation data into regions (according to different approaches) and, for each region, fits a (dispersion-adjusted) binomial regression model to regional methylation data, and reports the estimated smooth covariate effects and regional p-values for the test of DMRs (differentially methylation regions). Over or under dispersion across loci is accounted for in the model by the combination of a multiplicative dispersion parameter (or scale parameter) and a sample-specific random effect.

This method can deal with outcomes, i.e. the number of methylated reads in a region, that are contaminated by known false methylation calling rate (`p0`) and false non-methylation calling rate (`1-p1`).

The covariate effects are assumed to smoothly vary across genomic regions. In order to estimate them, the algorithm first represents the functional parameters by a linear combination of a set of restricted cubic splines (with dimension `n.k`), and a smoothness penalization term which depends on the smoothing parameters `lambdas` is also added to control smoothness. The estimation is performed by an iterated EM algorithm. Each M step constitutes an outer Newton's iteration to estimate smoothing parameters `lambdas` and an inner P-IRLS iteration to estimate spline coefficients `alpha` for the covariate effects. Currently, the computation in the M step depends the implementation of `gam()` in package `mgcv`.

Usage

```
runSOMNiBUS(
  dat,
  split = list(approach = "region"),
  min.cpgs = 50,
  max.cpgs = 2000,
  n.k,
  p0 = 0.003,
  p1 = 0.9,
  Quasi = TRUE,
  epsilon = 10^(-6),
  epsilon.lambda = 10^(-3),
  maxStep = 200,
  binom.link = "logit",
  method = "REML",
  covs = NULL,
  RanEff = TRUE,
  reml.scale = FALSE,
  scale = -2,
  verbose = TRUE
)
```

Arguments

`dat` a data frame with rows as individual CpGs appearing in all the samples. The first 4 columns should contain the information of `Meth_Counts` (methylated counts), `Total_Counts` (read depths), `Position` (Genomic position for the CpG site) and `ID` (sample ID). The covariate information, such as disease status or cell type composition, are listed in column 5 and onwards.

`split` this `list` must contain at least the element `approach` which corresponds to the partitioning approach used to split the data into independent regions. The partitioning methods available are:

- `"region"` (partitioning based on the spacing of CpGs),
- `"density"` (partitioning based on CpG density),
- `"chromatin"` (partitioning based on chromatin states),
- `"gene"` (partitioning based on gene regions),
- `"granges"` (partitioning based on user-specific annotations provided as a `GenomicRanges` object),
- `"bed"` (partitioning based on user-specific annotations provided in a BED file).

This list should also contain additional parameters specific to each partitioning approach (see the documentation of each approach for details).

`min.cpgs` positive integer defining the minimum number of CpGs within a region for the algorithm to perform optimally. The default value is 50.

`max.cpgs` positive integer defining the maximum number of CpGs within a region for the algorithm to perform optimally. The default value is 2000.

`n.k` a vector of basis dimensions for the intercept and individual covariates. `n.k` specifies an upper limit of the degrees of each functional parameters. The length of n.k should equal to the number of covariates plus 1 (for the intercept)). We recommend basis dimensions n.k, approximately equal to the number of unique CpGs in the region divided by 20. This parameter will be computed automatically, when several regions are generated by the partitioning function.

`p0` the probability of observing a methylated read when the underlying true status is unmethylated. `p0` is the rate of false methylation calls, i.e. false positive rate.

`p1` the probability of observing a methylated read when the underlying true status is methylated. `1-p1` is the rate of false non-methylation calls, i.e. false negative rate.

`Quasi` whether a Quasi-likelihood estimation approach will be used; in other words, whether a multiplicative dispersion is added in the model or not.

`epsilon` numeric; stopping criterion for the closeness of estimates of spline coefficients from two consecutive iterations.

`epsilon.lambda` numeric; stopping criterion for the closeness of estimates of smoothing parameter `lambda` from two consecutive iterations.

`maxStep` the algorithm will step if the iteration steps exceed `maxStep`.

`binom.link` the link function used in the binomial regression model; the default is the logit link.

method	the method used to estimate the smoothing parameters. The default is the 'REML' method which is generally better than prediction based criterion GCV.cp.
covs	a vector of covariate names. The covariates with names in covs will be included in the model and their covariate effects will be estimated. The default is to fit all covariates in dat
RanEff	whether sample-level random effects are added or not
reml.scale	whether a REML-based scale (dispersion) estimator is used. The default is Fletcher-based estimator.
scale	negative values mean scale parameter should be estimated; if a positive value is provided, a fixed scale will be used.
verbose	logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

This function returns a list of models (one by independent region) including objects:

- est: estimates of the spline basis coefficients alpha
- lambda: estimates of the smoothing parameters for each functional parameters
- est.pi: predicted methylation levels for each row in the input data
- ite.points: estimates of est, lambda at each EM iteration
- cov1: estimated variance-covariance matrix of the basis coefficients alphas
- reg.out: regional testing output obtained using Fletcher-based dispersion estimate; an additional 'ID' row would appear if RanEff is TRUE
- reg.out.reml.scale: regional testing output obtained using REML-based dispersion estimate;
- reg.out.gam: regional testing output obtained using (Fletcher-based) dispersion estimate from mgcv package;
- phi_fletcher: Fletcher-based estimate of the (multiplicative) dispersion parameter;
- phi_reml: REML-based estimate of the (multiplicative) dispersion parameter;
- phi_gam: Estimated dispersion parameter reported by mgcv;
- SE.out: a matrix of the estimated pointwise Standard Errors (SE); number of rows are the number of unique CpG sites in the input data and the number of columns equal to the total number of covariates fitted in the model (the first one is the intercept);
- SE.out.REML.scale: a matrix of the estimated pointwise Standard Errors (SE); the SE calculated from the REML-based dispersion estimates
- uni.pos: the genomic postions for each row of CpG sites in the matrix SE.out;
- Beta.out: a matrix of the estimated covariate effects beta(t), where t denotes the genomic positions;
- ncovs: number of functional paramters in the model (including the intercept);
- sigma00: estimated variance for the random effect if RanEff is TRUE; NA if RanEff is FALSE.

Author(s)

Audrey Lemaçon

Examples

```
#--------------------------------------------------------------#
data(RAdat)
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
outs <- runSOMNiBUS(
  dat=RAdat.f, split = list(approach = "region", gap = 1e6), min.cpgs = 5,
  n.k = rep(5,3), p0 = 0.003, p1 = 0.9
)
```

---

splitDataByBed *Split methylation data into regions based on the genomic annotations*

Description

This function splits the methylation data into regions based on the genomic annotation provided under the form of a 1-based BED file

Usage

```
splitDataByBed(
  dat,
  chr,
  bed,
  gap = -1,
  min.cpgs = 50,
  max.cpgs = 2000,
  verbose = TRUE
)
```

Arguments

dat	a data frame with rows as individual CpGs appearing in all the samples. The first 4 columns should contain the information of `Meth_Counts` (methylated counts), `Total_Counts` (read depths), `Position` (Genomic position for the CpG site) and `ID` (sample ID). The covariate information, such as disease status or cell type composition, are listed in column 5 and onwards.
chr	character vector containing the chromosome information. Its length should be equal to the number of rows in `dat`.
bed	character, path to the 1-based BED file containing the annotations

gap — integer defining the maximum gap that is allowed between two regions to be considered as overlapping. According to the `GenomicRanges::findOverlaps` function, the gap between 2 ranges is the number of positions that separate them. The gap between 2 adjacent ranges is 0. By convention when one range has its start or end strictly inside the other (i.e. non-disjoint ranges), the gap is considered to be -1. Decimal values will be rounded to the nearest integer. The default value is `-1` (meaning strict overlapping).

min.cpgs — positive integer defining the minimum number of CpGs within a region for the algorithm to perform optimally. The default value is 50.

max.cpgs — positive integer defining the maximum number of CpGs within a region for the algorithm to perform optimally. The default value is 2000.

verbose — logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

A named `list` of `data.frame` containing the data of each independent region.

Author(s)

Audrey Lemaçon

---

splitDataByChromatin — *Split methylation data into regions based on the chromatin states*

Description

This function splits the methylation data into regions based on the chromatin states predicted by ChromHMM software (Ernst and Kellis (2012)). The annotations come from the Bioconductor package `annnotatr`. Chromatin states determined by chromHMM are available in hg19 for nine cell lines (Gm12878, H1hesc, Hepg2, Hmec, Hsmm, Huvec, K562, Nhek, and Nhlf).

Usage

```
splitDataByChromatin(
  dat,
  chr,
  cell.line,
  states,
  gap = -1,
  min.cpgs = 50,
  max.cpgs = 2000,
  verbose = TRUE
)
```

Arguments

`dat` a data frame with rows as individual CpGs appearing in all the samples. The first 4 columns should contain the information of `Meth_Counts` (methylated counts), `Total_Counts` (read depths), `Position` (Genomic position for the CpG site) and `ID`(sample ID). The covariate information, such as disease status or cell type composition, are listed in column 5 and onwards.

`chr` character vector containing the chromosome information. Its length should be equal to the number of rows in `dat`.

`cell.line` character defining the cell line of interest. Nine cell lines are available:

- `"gm12878"`: Lymphoblastoid cells GM12878,
- `"h1hesc"`: Embryonic cells H1 hESC,
- `"hepg2"`: Liver carcinoma HepG2,
- `"hmec"`, Mammary epithelial cells HMEC,
- `"hsmm"`, Skeletal muscle myoblasts HSMM,
- `"huvec"`: Umbilical vein endothelial HUVEC,
- `"k562"`: Myelogenous leukemia K562,
- `"nhek"`: Keratinocytes NHEK,
- `"nhlf"`: Normal human lung fibroblasts NHLF.

`states` character vector defining the chromatin states of interest among the following available options:

- `"ActivePromoter"`: Active Promoter
- `"WeakPromoter"`: Weak Promoter
- `"PoisedPromoter"`: Poised Promoter
- `"StrongEnhancer"`: Strong Enhancer
- `"WeakEnhancer"`: Weak/poised Enhancer
- `"Insulator"`: Insulator
- `"TxnTransition"`: Transcriptional Transition
- `"TxnElongation"`: Transcriptional Elongation
- `"WeakTxn"`: Weak Transcribed
- `"Repressed"`: Polycomb-Repressed
- `"Heterochrom"`: Heterochromatin; low signal
- `"RepetitiveCNV"`: Repetitive/Copy Number Variation Use `state="all"` to select all the states simultaneously.

`gap` this integer defines the maximum gap that is allowed between two regions to be considered as overlapping. According to the `GenomicRanges::findOverlaps` function, the gap between 2 ranges is the number of positions that separate them. The gap between 2 adjacent ranges is 0. By convention when one range has its start or end strictly inside the other (i.e. non-disjoint ranges), the gap is considered to be -1. Decimal values will be rounded to the nearest integer. The default value is `-1`.

`min.cpgs` positive integer defining the minimum number of CpGs within a region for the algorithm to perform optimally. The default value is 50.

`max.cpgs`	positive integer defining the maximum number of CpGs within a region for the algorithm to perform optimally. The default value is 2000.
`verbose`	logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

A `list` of `data.frame` containing the data of each independent region.

Author(s)

Audrey Lemaçon

Examples

```
#------------------------------------------------------------#
data(RAdat)
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
results <- splitDataByChromatin(dat = RAdat.f,
cell.line = "huvec", chr = rep(x = "chr4", times = nrow(RAdat.f)),
states = "Insulator", verbose = FALSE)
```

`splitDataByDensity` *Split methylation data into regions based on the density of CpGs*

Description

This function splits the methylation data into regions based on the density of CpGs.

Usage

```
splitDataByDensity(
  dat,
  window.size = 100,
  by = 1,
  min.density = 5,
  gap = 10,
  min.cpgs = 50,
  max.cpgs = 2000,
  verbose = TRUE
)
```

Arguments

`dat`	a data frame with rows as individual CpGs appearing in all the samples. The first 4 columns should contain the information of `Meth_Counts` (methylated counts), `Total_Counts` (read depths), `Position` (Genomic position for the CpG site) and `ID` (sample ID). The covariate information, such as disease status or cell type composition, are listed in column 5 and onwards.
`window.size`	this positive integer defines the size of the sliding window in bp. Decimal values will be rounded to the nearest integer. The value should be greater than 10. The default value is `100` (100 bp)
`by`	positive integer defines by how many base pairs the window moves at each increment. Decimal values will be rounded to the nearest integer. The default value is `1` (1 bp).
`min.density`	positive integer defines the minimum density threshold for each window. Decimal values will be rounded to the nearest integer. The default value is `5` (5 CpGs/`window.size`).
`gap`	positive integer defining the gap width beyond which we consider that two regions are independent. Decimal values will be rounded to the nearest integer. The default value is `10` (10bp).
`min.cpgs`	positive integer defining the minimum number of CpGs within a region for the algorithm to perform optimally. The default value is 50.
`max.cpgs`	positive integer defining the maximum number of CpGs within a region for the algorithm to perform optimally. The default value is 2000.
`verbose`	logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

A named `list` of `data.frame` containing the data of each independent region.

Author(s)

Audrey Lemaçon

Examples

```
#--------------------------------------------------------------#
data(RAdat)
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
results <- splitDataByDensity(dat = RAdat.f, window.size = 100, by = 1,
min.density = 5, gap = 10, min.cpgs = 50, verbose = FALSE)
```

`splitDataByGene` *Split methylation data into regions based on the genes annotations*

Description

This function splits the methylation data into regions based on the genes. The annotations are coming from the Bioconductor package `annnotatr`.

Usage

```
splitDataByGene(
  dat,
  chr,
  organism = "human",
  build = "hg38",
  types = "promoter",
  gap = -1,
  min.cpgs = 50,
  max.cpgs = 2000,
  verbose = TRUE
)
```

Arguments

`dat` a data frame with rows as individual CpGs appearing in all the samples. The first 4 columns should contain the information of `Meth_Counts` (methylated counts), `Total_Counts` (read depths), `Position` (Genomic position for the CpG site) and `ID`(sample ID). The covariate information, such as disease status or cell type composition, are listed in column 5 and onwards.

`chr` character vector containing the chromosome information. Its length should be equal to the number of rows in `dat`.

`organism` character defining the organism of interest Only Homo sapiens (`"human"`) is available. Additional packages are required for Mus musculus (`"mouse"`), Rattus norvegicus (`"rat"`) and Drosophila melanogaster (`"fly"`). The matching is case-insensitive. The default value is `"human"`.

`build` character defining the version of the genome build on which the methylation data have been mapped. By default, the build is set to `"hg38"`, however the build `"hg19"` is also available for Homo sapiens: Once the additional packages are installed, the following organisms and builds are available:

- `"mm9"` and `"mm10"` for Mus musculus;
- `"rn4"`, `"rn5"` and `"rn6"` for Rattus norvegicus;
- `"dm3"` and `"dm6"` for Drosophila melanogaster;

`types` character vector defining the type of genic annotations to use among the following options:

- `"upstream"` for the annotations included 1-5Kb upstream of the TSS;

- "promoter" for the annotations included < 1Kb upstream of the TSS;
- "threeprime" for the annotations included in 3' UTR;
- "fiveprime" for the annotations included in the 5' UTR;
- "exon" for the annotations included in the exons;
- "intron" for the annotations included in the introns;
- "all" for all the annotations aforementioned. The default value is "promoter".

`gap` this integer defines the maximum gap allowed between two regions to be considered as overlapping. According to the GenomicRanges::findOverlaps function, the gap between 2 ranges is the number of positions that separate them. The gap between 2 adjacent ranges is 0. By convention when one range has its start or end strictly inside the other (i.e. non-disjoint ranges), the gap is considered to be -1. Decimal values will be rounded to the nearest integer. The default value is -1.

`min.cpgs` positive integer defining the minimum number of CpGs within a region for the algorithm to perform optimally. The default value is 50.

`max.cpgs` positive integer defining the maximum number of CpGs within a region for the algorithm to perform optimally. The default value is 2000.

`verbose` logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

A named list of data.frame containing the data of each independent region.

Author(s)

Audrey Lemaçon

Examples

```
#------------------------------------------------------------#
data(RAdat)
# Add a column containing the chromosome information
RAdat$Chr <- "chr4"
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
results <- splitDataByGene(dat = RAdat.f,
chr = rep(x = "chr1", times = nrow(RAdat.f)), verbose = FALSE)
```

---

splitDataByGRanges — *Split methylation data into regions based on the genomic annotations*

Description

This function splits the methylation data into regions based on the genomic annotations provided under the form of a GenomicRanges object.

Usage

```
splitDataByGRanges(
  dat,
  chr,
  annots,
  gap = -1,
  min.cpgs = 50,
  max.cpgs = 2000,
  verbose = TRUE
)
```

Arguments

`dat`	a data frame with rows as individual CpGs appearing in all the samples. The first 4 columns should contain the information of `Meth_Counts` (methylated counts), `Total_Counts` (read depths), `Position` (Genomic position for the CpG site) and `ID` (sample ID). The covariate information, such as disease status or cell type composition, are listed in column 5 and onwards.
`chr`	character vector containing the chromosome information. Its length should be equal to the number of rows in `dat`.
`annots`	GenomicRanges object containing the annotations
`gap`	integer defining the maximum gap that is allowed between two regions to be considered as overlapping. According to the `GenomicRanges::findOverlaps` function, the gap between 2 ranges is the number of positions that separate them. The gap between 2 adjacent ranges is 0. By convention when one range has its start or end strictly inside the other (i.e. non-disjoint ranges), the gap is considered to be -1. Decimal values will be rounded to the nearest integer. The default value is `-1` (meaning strict overlaping).
`min.cpgs`	positive integer defining the minimum number of CpGs within a region for the algorithm to perform optimally. The default value is 50.
`max.cpgs`	positive integer defining the maximum number of CpGs within a region for the algorithm to perform optimally. The default value is 2000.
`verbose`	logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

A named `list` of `data.frame` containing the data of each independent region.

Author(s)

Audrey Lemaçon

Examples

```
#--------------------------------------------------------------#
data(RAdat)
```

```
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
annot <- GenomicRanges::GRanges(seqnames = "chr1", IRanges::IRanges(
start = c(102711720,102711844,102712006,102712503,102712702),
end = c(102711757,102711909,102712195,102712637,102712712)
))
results <- splitDataByGRanges(dat = RAdat.f,
chr = rep(x = "chr1", times = nrow(RAdat.f)),
annots = annot, gap = -1, min.cpgs = 5)
```

`splitDataByRegion` *Split methylation data into regions based on the spacing of CpGs*

Description

This function splits the methylation data into regions based on the spacing of CpGs.

Usage

```
splitDataByRegion(
  dat,
  gap = 1e+06,
  min.cpgs = 50,
  max.cpgs = 2000,
  verbose = TRUE
)
```

Arguments

`dat`	a data frame with rows as individual CpGs appearing in all the samples. The first 4 columns should contain the information of `Meth_Counts` (methylated counts), `Total_Counts` (read depths), `Position` (Genomic position for the CpG site) and `ID` (sample ID). The covariate information, such as disease status or cell type composition, are listed in column 5 and onwards.
`gap`	positive integer defining the gap width beyond which we consider that two regions are independent. Odd and decimal values will be rounded to the next even numbers (e.g. 8.2 and 8.7 become gaps of 8 and 10 respectively). The default value is `1e+6` (1Mb).
`min.cpgs`	positive integer defining the minimum number of CpGs within a region for the algorithm to perform optimally. The default value is 50.
`max.cpgs`	positive integer defining the maximum number of CpGs within a region for the algorithm to perform optimally. The default value is 2000.
`verbose`	logical indicates if the algorithm should provide progress report information. The default value is TRUE.

Value

A named `list` of `data.frame` containing the data of each independent region.

Author(s)

Audrey Lemaçon

Examples

```
#-----------------------------------------------------------#
data(RAdat)
RAdat.f <- na.omit(RAdat[RAdat$Total_Counts != 0, ])
results <- splitDataByRegion( dat=RAdat.f, gap = 1e6, min.cpgs = 5,
verbose = FALSE)
```

Index