Package 'BPRMeth'

November 12, 2024

Type Package

Title Model higher-order methylation profiles

Version 1.32.0

Author Chantriolnt-Andreas Kapourani [aut, cre]

Maintainer Chantriolnt-Andreas Kapourani <kapouranis.andreas@gmail.com>

Description The BPRMeth package is a probabilistic method to quantify explicit features of methylation profiles, in a way that would make it easier to formally use such profiles in downstream modelling efforts, such as predicting gene expression levels or clustering genomic regions or cells according to their methylation profiles.

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 3.5.0), GenomicRanges

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports assertthat, methods, MASS, doParallel, parallel, e1071, earth, foreach, randomForest, stats, IRanges, S4Vectors, data.table, graphics, truncnorm, mvtnorm, Rcpp (>= 0.12.14), matrixcalc, magrittr, kernlab, ggplot2, cowplot, BiocStyle

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

biocViews ImmunoOncology, DNAMethylation, GeneExpression, GeneRegulation, Epigenetics, Genetics, Clustering, FeatureExtraction, Regression, RNASeq, Bayesian, KEGG, Sequencing, Coverage, SingleCell

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

git_branch RELEASE_3_20

git_last_commit 10da0eb

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-12

Contents

bernoulli_data *Synthetic Bernoulli data*

Description

A synthetic dataset containinig 300 entries from Bernoulli probit regression observations (e.g. single cell methylation data).

Usage

```
bernoulli_data
```

Format

A list with 300 elements, where each element element is an L x 2 matrix of observations, where:

1st column locations of obseravtions

2nd column methylation level, 0 - unmethylated, 1 - methylated

Value

Synthetic Bernoulli methylation data.

See Also

binomial_data, beta_data, gaussian_data

beta_data *Synthetic Beta data*

Description

A synthetic dataset containinig 300 entries from Beta probit regression observations (e.g. Beta values from array methylation data).

Usage

```
beta_data
```

Format

A list with 300 elements, where each element element is an L x 2 matrix of observations, where:

1st column locations of obseravtions

2nd column methylation level

Value

Synthetic Beta methylation data.

Details

The beta regression model is based on alternative parameterization of the beta density in terms of the mean and dispersion parameter:https://cran.r-project.org/web/packages/betareg/.

See Also

binomial_data, bernoulli_data, gaussian_data

---

binomial_data *Synthetic Binomial data*

Description

A synthetic dataset containinig 300 entries from Binomial probit regression observations (e.g. bulk methylation data).

Usage

```
binomial_data
```

Format

A list with 300 elements, where each element element is an L x 3 matrix of observations, where:

1st column locations of obseravtions

2nd column total trials

3rd column number of successes

Value

Synthetic Binomial methylation data.

See Also

bernoulli_data, beta_data, gaussian_data

---

boxplot_cluster_expr *Boxplot of clustered expression levels*

Description

Create a boxplot of clustered gene expression levels which depend on the clustered methylation profiles. Each colour denotes a different cluster.

Usage

```
boxplot_cluster_expr(cluster_obj, expr, anno, title = "Expression levels")
```

Arguments

cluster_obj	The output from cluster_profiles_vb or cluster_profiles_mle functions.
expr	The expression data object.
anno	The annotation data object.
title	Plot title

Value

A ggplot2 object.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

plot_cluster_profiles, plot_infer_profiles, plot_predicted_expr

Examples

```
# Cluster methylation profiles using 3 RBFs
basis <- create_rbf_object(M = 3)
# Perform clustering
cl_obj <- cluster_profiles_vb(X = encode_met$met, K = 3, model = "binomial",
           basis = basis, vb_max_iter = 5)
# Create plot
g <- boxplot_cluster_expr(cluster_obj = cl_obj, expr = encode_expr,
       anno = encode_met$anno)
```

BPRMeth *BPRMeth: Extracting higher order methylation features*

Description

Higher order methylation features for clustering and prediction in epigenomic studies

Usage

```
.datatable.aware
```

Format

An object of class logical of length 1.

Value

BPRMeth main package documentation.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

`bpr_cluster_wrap` *(DEPRECATED) Cluster methylation profiles*

Description

(DEPRECATED) `bpr_cluster_wrap` is a wrapper function that clusters methylation profiles using the EM algorithm. Initially, it performs parameter checking, and initializes main parameters, such as mixing proportions, basis function coefficients, then the EM algorithm is applied and finally model selection metrics are calculated, such as BIC and AIC.

Usage

```
bpr_cluster_wrap(
  x,
  K = 3,
  pi_k = NULL,
  w = NULL,
  basis = NULL,
  em_max_iter = 100,
  epsilon_conv = 1e-04,
  lambda = 1/2,
  opt_method = "CG",
  opt_itnmax = 100,
  init_opt_itnmax = 100,
  is_parallel = TRUE,
  no_cores = NULL,
  is_verbose = FALSE
)
```

Arguments

`x`	The binomial distributed observations, which has to be a list of elements of length N, where each element is an L x 3 matrix of observations, where 1st column contains the locations. The 2nd and 3rd columns contain the total trials and number of successes at the corresponding locations, repsectively. See `process_haib_caltech_wrap` on a possible way to get this data structure.
`K`	Integer denoting the number of clusters K.
`pi_k`	Mixing proportions.
`w`	A MxK matrix, where each column consists of the basis function coefficients for each corresponding cluster.
`basis`	A 'basis' object. E.g. see `create_rbf_object`.
`em_max_iter`	Integer denoting the maximum number of EM iterations.
`epsilon_conv`	Numeric denoting the convergence parameter for EM.
`lambda`	The complexity penalty coefficient for ridge regression.
`opt_method`	The optimization method to be used. See `optim` for possible methods. Default is "CG".
`opt_itnmax`	Optional argument giving the maximum number of iterations for the corresponding method. See `optim` for details.

init_opt_itnmax	Optimization iterations for obtaining the initial EM parameter values.
is_parallel	Logical, indicating if code should be run in parallel.
no_cores	Number of cores to be used, default is max_no_cores - 2.
is_verbose	Logical, print results during EM iterations.

Value

A 'bpr_cluster' object which, in addition to the input parameters, consists of the following variables:

- pi_k: Fitted mixing proportions.
- w: A MxK matrix with the fitted coefficients of the basis functions for each cluster k.
- NLL: The Negative Log Likelihood after the EM algorithm has finished.
- r_nk: Posterior probabilities of each promoter region belonging to each cluster.
- labels: Hard clustering assignments of each observation/promoter region.
- BIC: Bayesian Information Criterion metric.
- AIC: Akaike Information Criterion metric.
- ICL: Integrated Complete Likelihood criterion metric.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

Examples

```
ex_data <- meth_data
data_clust <- bpr_cluster_wrap(x = ex_data, em_max_iter = 3, opt_itnmax = 3,
                               init_opt_itnmax = 5, is_parallel = FALSE)
```

---

bpr_log_likelihood *Compute observation model log-likelihood*

Description

These functions evaluate the model log-likelihood and gradient for different observation models. Available models are "bpr" (i.e. "bernoulli" or "binomial"), "beta" and "lr" (i.e. "gaussian"). There are also functions to compute the sum and weighted sum of the observation model likelihoods, e.g. required for the EM algorithm. These functions are written in C++ for efficiency.

Usage

```
bpr_log_likelihood(w, X, H, lambda, is_nll)

bpr_gradient(w, X, H, lambda, is_nll)

betareg_log_likelihood(w, X, H, lambda, is_nll)

betareg_gradient(w, X, H, lambda, is_nll)
```

```
sum_weighted_bpr_lik(w, X_list, H_list, r_nk, lambda, is_nll)

sum_weighted_bpr_grad(w, X_list, H_list, r_nk, lambda, is_nll)

sum_weighted_betareg_lik(w, X_list, H_list, r_nk, lambda, is_nll)

sum_weighted_betareg_grad(w, X_list, H_list, r_nk, lambda, is_nll)

lr_log_likelihood(w, X, H, lambda = 0.5, is_nll = FALSE)
```

Arguments

w	A vector of parameters (i.e. coefficients of the basis functions)
X	An L X C matrix, where L are the total number of observations. The first column contains the input observations x (i.e. CpG locations). If "binomial" model then C=3, and 2nd and 3rd columns contain total number of trials and number of successes respectively. If "bernoulli" or "gaussian" model, then C=2 containing the output y (e.g. methylation level). if "beta" model, then C=3, where 2nd column contains output y and 3rd column the dispersion parameter. Each row corresponds to each row of the design matrix H.
H	The L x M matrix design matrix, where L is the number of observations and M the number of basis functions.
lambda	The complexity penalty coefficient for penalized regression.
is_nll	Logical, indicating if the Negative Log Likelihood should be returned.
X_list	A list of elements of length N, where each element is an L x K matrix of observations X.
H_list	A list of elements of length N, where each element contains the L x M design matrices H.
r_nk	A vector of length N containing the posterior probabilities (i.e. responsibilities) for each element of X_list.

Value

Returns the log likelihood or gradient of the observation model.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

eval_functions, infer_profiles_mle

bpr_optimize *(DEPRECATED) Optimize BPR negative log likelihood function*

Description

(DEPRECATED) The function bpr_optimize minimizes the negative log likelihood of the BPR function. Since it cannot be evaluated analytically, an optimization procedure is used. The optim packages is used for performing optimization.

Usage

```
bpr_optim(x, ...)

## S3 method for class 'list'
bpr_optim(
  x,
  w = NULL,
  basis = NULL,
  lambda = 1/2,
  opt_method = "CG",
  opt_itnmax = 100,
  is_parallel = TRUE,
  no_cores = NULL,
  ...
)

## S3 method for class 'matrix'
bpr_optim(
  x,
  w = NULL,
  basis = NULL,
  lambda = 1/2,
  opt_method = "CG",
  opt_itnmax = 100,
  ...
)
```

Arguments

x	The input object, either a matrix or a list.
...	Additional parameters.
w	A vector of parameters (i.e. coefficients of the basis functions)
basis	A 'basis' object. E.g. see create_rbf_object.
lambda	The complexity penalty coefficient for ridge regression.
opt_method	The optimization method to be used. See optim for possible methods. Default is "CG".
opt_itnmax	Optional argument giving the maximum number of iterations for the corresponding method. See optim for details.
is_parallel	Logical, indicating if code should be run in parallel.
no_cores	Number of cores to be used, default is max_no_cores - 2.

Value

Depending on the input object x:

- If x is a list: An object containing the following elements:
  - W_opt: An Nx(M+1) matrix with the optimized parameter values. Each row of the matrix corresponds to each element of the list x. The columns are of the same length as the parameter vector w (i.e. number of basis functions).
  - Mus: An N x M matrix with the RBF centers if basis object is create_rbf_object, otherwise NULL.
  - basis: The basis object.
  - w: The initial values of the parameters w.
- If x is a matrix: An object containing the following elements:
  - w_opt: Optimized values for the coefficient vector w. The length of the result is the same as the length of the vector w.
  - basis: The basis object.
- If calling bpr_optim_fast just the optimal weight matrix W_opt.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

create_basis, eval_functions

Examples

```
# Example of optimizing parameters for synthetic data using default values
data <- meth_data
out_opt <- bpr_optim(x = data, is_parallel = FALSE, opt_itnmax = 3)

#-------------------------------------

# Example of optimizing parameters for synthetic data using 3 RBFs
ex_data <- meth_data
basis <- create_rbf_object(M=3)
out_opt <- bpr_optim(x = ex_data, is_parallel = FALSE, basis = basis,
                     opt_itnmax = 3)

#-------------------------------------

# Example of of specific promoter region using 2 RBFs
basis <- create_rbf_object(M=2)
w <- c(0.1, 0.1, 0.1)
data <- meth_data[[1]]
out_opt <- bpr_optim(x = data, w = w, basis = basis, fit_feature = "NLL",
                     opt_itnmax = 3)
```

bpr_predict_wrap *(DEPRECATED) Predict gene expression from methylation profiles*

Description

(DEPRECATED) bpr_predict_wrap is a function that wraps all the necessary subroutines for performing prediction on gene expression levels. Initially, it optimizes the parameters of the basis functions so as to learn the methylation profiles. Then, uses the learned parameters / coefficients of the basis functions as input features for performing regression in order to predict the corresponding gene expression levels.

Usage

```
bpr_predict_wrap(
  formula = NULL,
  x,
  y,
  model_name = "svm",
  w = NULL,
  basis = NULL,
  train_ind = NULL,
  train_perc = 0.7,
  fit_feature = "RMSE",
  cov_feature = TRUE,
  opt_method = "CG",
  opt_itnmax = 100,
  is_parallel = TRUE,
  no_cores = NULL,
  is_summary = TRUE
)
```

Arguments

formula	An object of class formula, e.g. see lm function. If NULL, the simple linear regression model is used.
x	The binomial distributed observations, which has to be a list of elements of length N, where each element is an L x 3 matrix of observations, where 1st column contains the locations. The 2nd and 3rd columns contain the total trials and number of successes at the corresponding locations, repsectively. See process_haib_caltech_wrap on a possible way to get this data structure.
y	Corresponding gene expression data for each element of the list x.
model_name	A string denoting the regression model. Currently, available models are: "svm", "randomForest", "rlm", "mars" and "lm".
w	Optional vector of initial parameter / coefficient values.
basis	Optional basis function object, default is an 'rbf' object, see create_rbf_object.
train_ind	Optional vector containing the indices for the train set.
train_perc	Optional parameter for defining the percentage of the dataset to be used for training set, the remaining will be the test set.

fit_feature	Return additional feature on how well the profile fits the methylation data. Either NULL for ignoring this feature or one of the following: 1) "RMSE" for returning the fit of the profile using the RMSE as measure of error or 2) "NLL" for returning the fit of the profile using the Negative Log Likelihood as measure of error.
cov_feature	Logical, whether to return an additional feature for the CpG coverage across the promoter region.
opt_method	The optimization method to be used. See optim for possible methods. Default is "CG".
opt_itnmax	Optional argument giving the maximum number of iterations for the corresponding method. See optim for details.
is_parallel	Logical, indicating if code should be run in parallel.
no_cores	Number of cores to be used, default is max_no_cores - 2.
is_summary	Logical, print the summary statistics.

Value

A 'bpr_predict' object which, in addition to the input parameters, consists of the following variables:

- W_opt: An Nx(M+1) matrix with the optimized parameter values. Each row of the matrix corresponds to each element of the list x. The columns are of the same length as the parameter vector w (i.e. number of basis functions).
- Mus: An N x M matrix with the RBF centers if basis object is create_rbf_object, otherwise NULL.
- train: The training data.
- test: The test data.
- gex_model: The fitted regression model.
- train_pred The predicted values for the training data.
- test_pred The predicted values for the test data.
- train_errors: The training error metrics.
- test_errors: The test error metrics.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

bpr_optimize, create_basis

Examples

```
obs <- meth_data
y   <- gex_data
basis <- create_rbf_object(M = 5)
out   <- bpr_predict_wrap(x = obs, y = y, basis = basis,
                          is_parallel = FALSE, opt_itnmax = 3)
```

`cluster_profiles_mle` *Cluster methylation profiles using EM*

Description

General purpose functions for clustering latent profiles for different observation models using maximum likelihood estimation (MLE) and the EM algorithm. Initially, it performs parameter checking, and initializes main parameters, such as mixing proportions, basis function coefficients, then the EM algorithm is applied and finally model selection metrics are calculated, such as BIC and AIC.

Usage

```
cluster_profiles_mle(
  X,
  K = 3,
  model = NULL,
  basis = NULL,
  H = NULL,
  pi_k = NULL,
  lambda = 0.5,
  beta_dispersion = 5,
  gaussian_sigma = rep(0.2, K),
  w = NULL,
  em_max_iter = 50,
  epsilon_conv = 1e-04,
  opt_method = "CG",
  opt_itnmax = 50,
  init_opt_itnmax = 30,
  is_parallel = FALSE,
  no_cores = NULL,
  is_verbose = FALSE,
  ...
)
```

Arguments

`X`	The input data, which has to be a `list` of elements of length N, where each element is an `L X C` matrix, where L are the total number of observations. The first column contains the input observations x (i.e. CpG locations). If "binomial" model then C=3, and 2nd and 3rd columns contain total number of trials and number of successes respectively. If "bernoulli" or "gaussian" model, then C=2 containing the output y (e.g. methylation level). If "beta" model, then C=3, where 2nd column contains output y and 3rd column the dispersion parameter.
`K`	Integer denoting the total number of clusters K.
`model`	Observation model name as character string. It can be either 'bernoulli', 'binomial', 'beta' or 'gaussian'.
`basis`	A 'basis' object. E.g. see `create_basis`. If NULL, will an RBF object will be created.
`H`	Optional, design matrix of the input data X. If NULL, H will be computed inside the function.

pi_k	Vector of length K, denoting the mixing proportions.
lambda	The complexity penalty coefficient for ridge regression.
beta_dispersion	Dispersion parameter, only used for Beta distribution and will be the same for all observations.
gaussian_sigma	Initial standard deviation of the noise term, only used when having "gaussian" observation model.
w	Optional, an (M+1)xK matrix of the initial parameters, where each column consists of the basis function coefficients for each corresponding cluster k. If NULL, will be assigned with default values.
em_max_iter	Integer denoting the maximum number of EM iterations.
epsilon_conv	Numeric denoting the convergence threshold for EM.
opt_method	The optimization method to be used. See optim for possible methods. Default is "CG".
opt_itnmax	Optional argument giving the maximum number of iterations for the corresponding method. See optim for details.
init_opt_itnmax	Optimization iterations for obtaining the initial EM parameter values.
is_parallel	Logical, indicating if code should be run in parallel.
no_cores	Number of cores to be used, default is max_no_cores - 1.
is_verbose	Logical, print results during EM iterations.
...	Additional parameters.

Value

An object of class cluster_profiles_mle_"obs_model" with the following elements:

- W: An (M+1) X K matrix with the optimized parameter values for each cluster. Each column of the matrix corresponds a different cluster k. M are the number of basis functions.
- pi_k: Mixing proportions.
- r_nk: An (N X K) responsibility matrix of each observations being explained by a specific cluster.
- basis: The basis object.
- nll: The negative log likelihood vector.
- labels: Cluster assignment labels.
- bic: Bayesian Information Criterion metric.
- aic: Akaike Information Criterion metric.
- icl: Integrated Complete Likelihood criterion metric.
- gaussian_sigma: Optimized standard deviation for gaussian observation model.

Details

The beta regression model is based on alternative parameterization of the beta density in terms of the mean and dispersion parameter: https://cran.r-project.org/web/packages/betareg/. For modelling details for Binomial/Bernoulli observation model check the paper for BPRMeth: https://academic.oup.com/bioinformatics/article/32/17/i405/2450762 .

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

create_basis, cluster_profiles_vb infer_profiles_vb, infer_profiles_mle, infer_profiles_gibbs, create_region_object

Examples

```
# Example of optimizing parameters for synthetic data using 3 RBFs

basis <- create_rbf_object(M=3)
out <- cluster_profiles_mle(X = binomial_data, model = "binomial",
  basis=basis, em_max_iter = 5, opt_itnmax = 5, init_opt_itnmax=5,
  is_parallel = FALSE)

#-------------------------------------

basis <- create_rbf_object(M=3)
out <- cluster_profiles_mle(X = gaussian_data, model = "gaussian",
  basis=basis, em_max_iter = 5, opt_itnmax = 5, init_opt_itnmax=5,
  is_parallel = FALSE)
```

---

cluster_profiles_vb *Cluster methylation profiles using VB*

Description

General purpose functions for clustering latent profiles for different observation models using Variational Bayes (VB) EM-like algorithm.

Usage

```
cluster_profiles_vb(
  X,
  K = 3,
  model = NULL,
  basis = NULL,
  H = NULL,
  delta_0 = NULL,
  w = NULL,
  gaussian_l = 50,
  alpha_0 = 0.5,
  beta_0 = 0.1,
  vb_max_iter = 100,
  epsilon_conv = 1e-04,
  is_verbose = FALSE,
  ...
)
```

Arguments

X	The input data, which has to be a list of elements of length N, where each element is an L X C matrix, where L are the total number of observations. The first column contains the input observations x (i.e. CpG locations). If "binomial" model then C=3, and 2nd and 3rd columns contain total number of trials and number of successes respectively. If "bernoulli" or "gaussian" model, then C=2 containing the output y (e.g. methylation level).
K	Integer denoting the total number of clusters K.
model	Observation model name as character string. It can be either 'bernoulli', 'binomial', 'beta' or 'gaussian'.
basis	A 'basis' object. E.g. see create_basis. If NULL, will an RBF object will be created.
H	Optional, design matrix of the input data X. If NULL, H will be computed inside the function.
delta_0	Parameter vector of the Dirichlet prior on the mixing proportions pi.
w	Optional, an (M+1)xK matrix of the initial parameters, where each column consists of the basis function coefficients for each corresponding cluster k. If NULL, will be assigned with default values.
gaussian_l	Noise precision parameter, only used when having "gaussian" observation model.
alpha_0	Hyperparameter: shape parameter for Gamma distribution. A Gamma distribution is used as prior for the precision parameter tau.
beta_0	Hyperparameter: rate parameter for Gamma distribution. A Gamma distribution is used as prior for the precision parameter tau.
vb_max_iter	Integer denoting the maximum number of VB iterations.
epsilon_conv	Numeric denoting the convergence threshold for VB.
is_verbose	Logical, print results during VB iterations.
...	Additional parameters.

Value

An object of class cluster_profiles_vb_"obs_model" with the following elements:

- W: An (M+1) X K matrix with the optimized parameter values for each cluster, M are the number of basis functions. Each column of the matrix corresponds a different cluster k.
- W_Sigma: A list with the covariance matrices of the posterior parmateter W for each cluster k.
- r_nk: An (N X K) responsibility matrix of each observations being explained by a specific cluster.
- delta: Optimized Dirichlet paramter for the mixing proportions.
- alpha: Optimized shape parameter of Gamma distribution.
- beta: Optimized rate paramter of the Gamma distribution
- basis: The basis object.
- lb: The lower bound vector.
- labels: Cluster assignment labels.
- pi_k: Expected value of mixing proportions.

Details

The modelling and mathematical details for clustering profiles using mean-field variational inference are explained here: http://rpubs.com/cakapourani/ . More specifically:

- For Binomial/Bernoulli observation model check: http://rpubs.com/cakapourani/vb-mixture-bpr
- For Gaussian observation model check: http://rpubs.com/cakapourani/vb-mixture-lr

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

create_basis, cluster_profiles_mle infer_profiles_vb, infer_profiles_mle, infer_profiles_gibbs, create_region_object

Examples

```
# Example of optimizing parameters for synthetic data using 3 RBFs
basis <- create_rbf_object(M=3)
out <- cluster_profiles_vb(X = binomial_data, model = "binomial",
  basis=basis, vb_max_iter = 10)

#-------------------------------------

basis <- create_rbf_object(M=3)
out <- cluster_profiles_vb(X = gaussian_data, model = "gaussian",
  basis=basis, vb_max_iter = 10)
```

---

create_anno_region *Create annotation regions*

Description

create_anno_region creates annotation regions from annotation data, using the central point of the annotation features as ground truth labels we create genomic regions N bp upstream and M bp downstream of central location.

Usage

```
create_anno_region(
  anno,
  chrom_size = NULL,
  is_centre = FALSE,
  is_window = TRUE,
  upstream = -5000,
  downstream = 5000
)
```

Arguments

anno	A GRanges object containing the annotation data, this normally would be the output from read_anno function.
chrom_size	Object containing genome chromosome sizes, normally would be the output of read_chrom_size function.
is_centre	Logical, whether 'start' and 'end' locations are pre-centred. If TRUE, the mean of the locations will be chosen as centre. If FALSE, the 'start' will be chosen as the center; e.g. for genes the 'start' denotes the TSS and we use this as centre to obtain K-bp upstream and downstream of TSS.
is_window	Whether to consider a predefined window region around centre. If TRUE, then 'upstream' and 'downstream' parameters are used, otherwise we consider the whole region from start to end location.
upstream	Integer defining the length of bp upstream of 'centre' for creating the genomic region. If is_window = FALSE, this parameter is ignored.
downstream	Integer defining the length of bp downstream of 'centre' for creating the genomic region. If is_window = FALSE, this parameter is ignored.

Value

A GRanges object containing the genomic regions.

The GRanges object contains two or three additional metadata column:

- id: Genomic region id.
- centre: Central location of each genomic region.
- name: (Optional) Genomic region name.

This column can be accessed as follows: granges_object$tss

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

create_region_object, read_anno

Examples

```
# Obtain the path to files
file <- system.file("extdata", "dummy_anno.bed", package = "BPRMeth")
anno_dt <- read_anno(file, is_anno_region = FALSE)
# Create genomic region
gen_region <- create_anno_region(anno_dt)
# Extract ID
id <- gen_region$id
```

create_basis *Create basis objects*

Description

These functions create different basis objects, which can be used as input to complex functions in order to perform computations depending on the class of the basis function.

Usage

```
create_rbf_object(
  M = 2,
  gamma = NULL,
  mus = NULL,
  eq_spaced_mus = TRUE,
  whole_region = TRUE
)

create_polynomial_object(M = 1)

create_fourier_object(M = 2, period = 2)
```

Arguments

M	The number of the basis functions. In case of Fourier basis, this number should be even, since we need to have pairs of sines and cosines and the constant term is added by default.
gamma	Inverse width of radial basis function.
mus	Optional centers of the RBF.
eq_spaced_mus	Logical, if TRUE, equally spaced centers are created, otherwise centers are created using kmeans algorithm.
whole_region	Logical, indicating if the centers will be evaluated equally spaced on the whole region, or between the min and max of the observation values.
period	The period, that is the basis functions are periodic on a specific interval. Best choice is the range of the points used for regression.

Value

A basis object of class 'rbf', 'polynomial' or 'fourier'.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

eval_functions, design_matrix

Examples

```
(obj <- create_rbf_object(M = 2))

#--------------------------------

(obj <- create_polynomial_object(M = 2))

(obj <- create_fourier_object(M = 2, period = 1))
```

`create_region_object` *Create genomic region data*

Description

`create_region_object` creates genomic regions (e.g. forms methylation regions data) using as input methylation and annotation data with genomic regions of interest.

Usage

```
create_region_object(
  met_dt,
  anno_dt,
  cov = 5,
  sd_thresh = 0.1,
  ignore_strand = TRUE,
  filter_empty_region = TRUE,
  fmin = -1,
  fmax = 1
)
```

Arguments

`met_dt`	A `GRanges` object with methylation data, whose format should be similar to `read_met` function.
`anno_dt`	A `GRanges` object with annotation data, whose format should be similar to `read_anno`.
`cov`	Integer defining the minimum coverage of CpGs that each region must contain.
`sd_thresh`	Optional numeric defining the minimum standard deviation of the methylation change in a region. This is used to filter regions with no methylation variability.
`ignore_strand`	Logical, whether or not to ignore strand information.
`filter_empty_region`	Logical, whether to discard genomic regions that have no CpG coverage or do not pass filtering options.
`fmin`	Minimum range value for location scaling. Under this version, it should be left to its default value -1.
`fmax`	Maximum range value for location scaling. Under this version, it should be left to its default value 1.

Value

A `list` object containing the two elements:

- `met`: A list containing methylation region data, where each entry in the list is an $L_i XD$ dimensional matrix, where L_i denotes the number of CpGs found in region `i`. The columns contain the following information:
  1. 1st column: Contains the locations of CpGs relative to centre. Note that the actual locations are scaled to the (fmin, fmax) region.
  2. 2nd column: If "bulk" data (i.e. binomial) it contains the total number of reads at each CpG location, otherwise the methylation level.
  3. 3rd column: If "bulk" data, the methylated reads at each CpG location, otherwise this D = 2 and this column is absent.

  . Rownames of each matrix contain the actual CpG genomic coordinates as <chr>:<location>.
- `anno`: The annotation object.

Note: The lengths of `met` and `anno` should match.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

`read_met`, `read_anno`

Examples

```
## Not run:
# Download the files and change the working directory to that location
met_dt <- read_met("name_of_met_file")
anno_dt <- read_anno("name_of_anno_file")

obj <- create_region_object(met_dt, anno_dt)

# Extract methylation regions
met <- obj$met

## End(Not run)
```

`design_matrix` *Generic function for creating design matrices*

Description

These functions call the appropriate methods depending on the class of the object `obj` to create RBF, polynomial or Fourier design matrices.

Usage

```
design_matrix(obj, ...)

## Default S3 method:
design_matrix(obj, ...)

## S3 method for class 'polynomial'
design_matrix(obj, obs, ...)

## S3 method for class 'rbf'
design_matrix(obj, obs, ...)

## S3 method for class 'fourier'
design_matrix(obj, obs, ...)
```

Arguments

obj	A basis function object.
...	Additional parameters.
obs	A vector of observations.

Value

A design matrix object

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

create_basis, eval_functions

Examples

```
obj <- create_polynomial_object(M=2)
obs <- c(0,.2,.5)
poly <- design_matrix(obj, obs)

#----------------

obj <- create_rbf_object(M=2)
obs <- c(0,.2,.5)
rbf <- design_matrix(obj, obs)

#----------------

obj <- create_fourier_object(M=2)
obs <- c(0,.2,.5)
fourier <- design_matrix(obj, obs)
```

encode_expr *Processed ENCODE expression data*

Description

Small subset of ENCODE expression data already in pre-processed format, which are used as a case study for the vignette.

Usage

```
encode_expr
```

Format

Expression data in format as the output of `read_expr` fucntion, i.e. a `data.table` with two columns: "id", "expr".

Value

Encode expression data

See Also

`create_region_object`, `read_met`, `read_anno`, `read_expr`

encode_met *Processed ENCODE methylation data*

Description

Small subset of ENCODE methylation data already in pre-processed format, which are used as a case study for the vignette.

Usage

```
encode_met
```

Format

A list object containing methylation regions and annotation data. This in general would be the output of the `create_region_object` function. It has the following two objects:

- `met`: A list containing the methylation regions, each element of the list is a different genomic region.
- `anno`: Corresponding annotation data.

Value

Encode methylation data

See Also

`create_region_object`, `read_met`, `read_anno`, `read_expr`

`eval_functions` *Evaluate basis functions*

Description

Method for evaluating an M basis function model with observation data `obs` and coefficients `w`.

Usage

```
eval_probit_function(obj, ...)

eval_function(obj, ...)

## S3 method for class 'rbf'
eval_function(obj, obs, w, ...)

## S3 method for class 'polynomial'
eval_function(obj, obs, w, ...)

## S3 method for class 'fourier'
eval_function(obj, obs, w, ...)
```

Arguments

`obj`	The basis function object.
`...`	Optional additional parameters
`obs`	Observation data.
`w`	Vector of length M, containing the coefficients of an M^{th}-order basis function.

Value

The evaluated function values.

NOTE that the `eval_probit_function` computes the probit transformed basis function values.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

`create_basis`, `design_matrix`

Examples

```
# Evaluate the probit transformed basis function values
obj <- create_rbf_object(M=2)
obs <- c(1,2,3)
w <- c(0.1, 0.3, -0.6)
out <- eval_probit_function(obj, obs, w)

# -------------------------
```

```
# Evaluate the RBF basis function values
obj <- create_rbf_object(M=2, mus = c(2,2.5))
obs <- c(1,2,3)
w <- c(0.1, 0.3, -0.6)
out <- eval_function(obj, obs, w)

# -------------------------

# Evaluate the Polynomial basis function values
obj <- create_polynomial_object(M=2)
obs <- c(1,2,3)
w <- c(0.1, 0.3, -0.6)
out <- eval_function(obj, obs, w)

# -------------------------

# Evaluate the Fourier basis function values
obj <- create_fourier_object(M=2)
obs <- c(1,2,3)
w <- c(0.1, 0.3, -0.6)
out <- eval_function(obj, obs, w)
```

`gaussian_data` *Synthetic Gaussian data*

Description

A synthetic dataset containinig 300 entries from Gaussian (i.e. linear) regression observations (e.g. M-values from array methylation data).

Usage

```
gaussian_data
```

Format

A list with 300 elements, where each element element is an L x 2 matrix of observations, where:

1st column locations of obseravtions

2nd column methylation level

Value

Synthetic Bernoulli methylation data.

See Also

`binomial_data`, `beta_data`, `bernoulli_data`

gex_data *Synthetic expression data*

Description

Corresponding gene expression data for the meth_data

Usage

```
gex_data
```

Format

A vector of length 600

Value

Synthetic gene expression data

See Also

meth_data, bernoulli_data, binomial_data, beta_data, gaussian_data

impute_bulk_met *Impute/predict bulk methylation states*

Description

Make predictions of missing methylation states, i.e. perfrom imputation using BPRmeth This requires keepin a subset of data as a held out test set during BPRMeth inference or providing a different file that contains chromosome and CpG locations.

Usage

```
impute_bulk_met(obj, anno, test_data = NULL, return_test = FALSE)
```

Arguments

obj	Output of BPRMeth inference object.
anno	A GRanges object with annotation data, whose format should be similar to read_anno.
test_data	Test data to evaluate performance.
return_test	Whether or not to return a list with the predictions.

Value

A list containing two vectors, the true methylation state and the predicted/imputed methylation states.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

partition_bulk_dataset

Examples

```
# Extract synthetic data
dt <- encode_met

# Partition to train and test set
dt <- partition_bulk_dataset(dt)

# Create basis object
basis_obj <- create_rbf_object(M = 3)

# Run BPRMeth
fit <- infer_profiles_mle(X = dt$met, model = "binomial",
   basis = basis_obj, is_parallel = FALSE, opt_itnmax = 10)

# Perform imputation
imputation_obj <- impute_bulk_met(obj = fit, anno = dt$anno,
                                  test_data = dt$met_test)
```

---

infer_profiles_gibbs *Infer methylation profiles using Gibbs sampling*

Description

General purpose functions for inferring latent profiles for different observation models using Gibbs sampling. Currently implemented observation models are: 'bernoulli' and 'binomial' and the auxiliary variable approach is used.

Usage

```
infer_profiles_gibbs(
  X,
  model = NULL,
  basis = NULL,
  H = NULL,
  w = NULL,
  mu_0 = NULL,
  cov_0 = NULL,
  gibbs_nsim = 500,
  gibbs_burn_in = 100,
  store_gibbs_draws = FALSE,
  is_parallel = FALSE,
  no_cores = NULL,
  ...
)
```

Arguments

X	The input data, either a matrix or a list of elements of length N, where each element is an L X C matrix, where L are the total number of observations. The first column contains the input observations x (i.e. CpG locations). If "binomial" model then C=3, and 2nd and 3rd columns contain total number of trials and number of successes respectively. If "bernoulli" then C=2 containing the output y (e.g. methylation level).
model	Observation model name as character string. It can be either 'bernoulli' or 'binomial'.
basis	A 'basis' object. E.g. see create_basis. If NULL, will an RBF object will be created.
H	Optional, design matrix of the input data X. If NULL, H will be computed inside the function.
w	A vector of initial parameters (i.e. coefficients of the basis functions). If NULL, it will be initialized inside the function.
mu_0	The prior mean hyperparameter vector for w.
cov_0	The prior covariance hyperparameter matrix for w.
gibbs_nsim	Total number of simulations for the Gibbs sampler.
gibbs_burn_in	Burn in period of the Gibbs sampler.
store_gibbs_draws	Logical indicating if we should keep the whole MCMC chain for further analysis.
is_parallel	Logical, indicating if code should be run in parallel.
no_cores	Number of cores to be used, default is max_no_cores - 1.
...	Additional parameters.

Value

An object of class infer_profiles_gibbs_"obs_model" with the following elements:

- W: An Nx(M+1) matrix with the posterior mean of the parameters w. Each row of the matrix corresponds to each element of the list X; if X is a matrix, then N = 1. The columns are of the same length as the parameter vector w (i.e. number of basis functions).
- W_sd: An Nx(M+1) matrix with the posterior standard deviation (sd) of the parameters W.
- basis: The basis object.
- nll_feat: NLL fit feature.
- rmse_feat: RMSE fit feature.
- coverage_feat: CpG coverage feature.
- W_draws: Optional, draws of the Gibbs sampler.

Details

The modelling and mathematical details for inferring profiles using Gibbs sampling are explained here: http://rpubs.com/cakapourani/ . More specifically:

- For Binomial observation model check: http://rpubs.com/cakapourani/bayesian-bpr-model
- For Bernoulli observation model check: http://rpubs.com/cakapourani/bayesian-bpr-model

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

create_basis, infer_profiles_mle, infer_profiles_vb, create_region_object

Examples

```
# Example of inferring parameters for synthetic data using 3 RBFs
basis <- create_rbf_object(M=3)
out <- infer_profiles_gibbs(X = binomial_data, model = "binomial",
   basis = basis, is_parallel = FALSE, gibbs_nsim = 10, gibbs_burn_in = 5)

#-------------------------------------
```

---

infer_profiles_mle *Infer methylation profiles using MLE*

Description

General purpose functions for inferring latent profiles for different observation models using maximum likelihood estimation (MLE). Current observation models are: 'bernoulli', 'binomial', 'beta' or 'gaussian'. For most models we cannot obtain an analytically tractable solution, hence an optimization procedure is used. The optim package is used for performing optimization.

Usage

```
infer_profiles_mle(
  X,
  model = NULL,
  basis = NULL,
  H = NULL,
  lambda = 0.5,
  w = NULL,
  beta_dispersion = 5,
  opt_method = "CG",
  opt_itnmax = 100,
  is_parallel = FALSE,
  no_cores = NULL,
  ...
)
```

Arguments

X	The input data, either a matrix or a list of elements of length N, where each element is an L X C matrix, where L are the total number of observations. The first column contains the input observations x (i.e. CpG locations). If "binomial" model then C=3, and 2nd and 3rd columns contain total number of trials and number of successes respectively. If "bernoulli" or "gaussian" model, then C=2 containing the output y (e.g. methylation level). If "beta" model, then C=3, where 2nd column contains output y and 3rd column the dispersion parameter.

model	Observation model name as character string. It can be either 'bernoulli', 'binomial', 'beta' or 'gaussian'.
basis	A 'basis' object. E.g. see create_basis. If NULL, will an RBF object will be created.
H	Optional, design matrix of the input data X. If NULL, H will be computed inside the function.
lambda	The complexity penalty coefficient for ridge regression.
w	A vector of initial parameters (i.e. coefficients of the basis functions).
beta_dispersion	Dispersion parameter, only used for Beta distribution and will be the same for all observations.
opt_method	The optimization method to be used. See optim for possible methods. Default is "CG".
opt_itnmax	Optional argument giving the maximum number of iterations for the corresponding method. See optim for details.
is_parallel	Logical, indicating if code should be run in parallel.
no_cores	Number of cores to be used, default is max_no_cores - 1.
...	Additional parameters.

Value

An object of class infer_profiles_mle_"obs_model" with the following elements:

- W: An Nx(M+1) matrix with the optimized parameter values. Each row of the matrix corresponds to each element of the list X; if X is a matrix, then N = 1. The columns are of the same length as the parameter vector w (i.e. number of basis functions).
- basis: The basis object.
- nll_feat: NLL fit feature.
- rmse_feat: RMSE fit feature.
- coverage_feat: CpG coverage feature.

Details

The beta regression model is based on alternative parameterization of the beta density in terms of the mean and dispersion parameter: https://cran.r-project.org/web/packages/betareg/ . For modelling details for Binomial/Bernoulli observation model check the paper for BPRMeth: https://academic.oup.com/bioinformatics/article/32/17/i405/2450762 .

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

create_basis, infer_profiles_vb, infer_profiles_gibbs, create_region_object

Examples

```
# Example of optimizing parameters for synthetic data using 3 RBFs
basis <- create_rbf_object(M=3)
out <- infer_profiles_mle(X = binomial_data, model = "binomial",
   basis = basis, is_parallel = FALSE, opt_itnmax = 10)

#-------------------------------------

basis <- create_rbf_object(M=3)
out <- infer_profiles_mle(X = beta_data, model = "beta",
   basis = basis, is_parallel = FALSE, opt_itnmax = 10)

#-------------------------------------

basis <- create_rbf_object(M=3)
out <- infer_profiles_mle(X = gaussian_data[[1]], model = "gaussian",
   basis = basis, is_parallel = FALSE, opt_itnmax = 10)
```

`infer_profiles_vb` *Infer methylation profiles using VB*

Description

General purpose functions for inferring latent profiles for different observation models using Variational Bayes (VB). Current observation models are: 'bernoulli', 'binomial' or 'gaussian'.

Usage

```
infer_profiles_vb(
  X,
  model = NULL,
  basis = NULL,
  H = NULL,
  w = NULL,
  gaussian_l = 50,
  alpha_0 = 0.5,
  beta_0 = 0.1,
  vb_max_iter = 100,
  epsilon_conv = 1e-05,
  is_parallel = FALSE,
  no_cores = NULL,
  is_verbose = FALSE,
  ...
)
```

Arguments

`X`	The input data, either a `matrix` or a `list` of elements of length N, where each element is an `L X C` matrix, where L are the total number of observations. The first column contains the input observations x (i.e. CpG locations). If "binomial" model then C=3, and 2nd and 3rd columns contain total number of trials and

	number of successes respectively. If "bernoulli" or "gaussian" model, then C=2 containing the output y (e.g. methylation level).
model	Observation model name as character string. It can be either 'bernoulli', 'binomial', 'beta' or 'gaussian'.
basis	A 'basis' object. E.g. see create_basis. If NULL, will an RBF object will be created.
H	Optional, design matrix of the input data X. If NULL, H will be computed inside the function.
w	A vector of initial parameters (i.e. coefficients of the basis functions). If NULL, it will be initialized inside the function.
gaussian_l	Noise precision parameter, only used when having "gaussian" observation model.
alpha_0	Hyperparameter: shape parameter for Gamma distribution. A Gamma distribution is used as prior for the precision parameter tau.
beta_0	Hyperparameter: rate parameter for Gamma distribution. A Gamma distribution is used as prior for the precision parameter tau.
vb_max_iter	Integer denoting the maximum number of VB iterations.
epsilon_conv	Numeric denoting the convergence threshold for VB.
is_parallel	Logical, indicating if code should be run in parallel.
no_cores	Number of cores to be used, default is max_no_cores - 1.
is_verbose	Logical, print results during VB iterations.
...	Additional parameters.

Value

An object of class infer_profiles_vb_"obs_model" with the following elements:

- W: An Nx(M+1) matrix with the optimized parameter values. Each row of the matrix corresponds to each element of the list X; if X is a matrix, then N = 1. The columns are of the same length as the parameter vector w (i.e. number of basis functions).
- W_Sigma: A list with covariance matrices for each element row in W.
- basis: The basis object.
- nll_feat: NLL fit feature.
- rmse_feat: RMSE fit feature.
- coverage_feat: CpG coverage feature.
- lb_feat: Lower Bound feature.

Details

The modelling and mathematical details for inferring profiles using mean-field variational inference are explained here: http://rpubs.com/cakapourani/ . More specifically:

- For Binomial/Bernoulli observation model check: http://rpubs.com/cakapourani/variational-bayes-bpr
- For Gaussian observation model check: http://rpubs.com/cakapourani/variational-bayes-lr

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

create_basis, infer_profiles_mle, predict_expr, create_region_object

Examples

```
# Example of inferring parameters for synthetic data using 3 RBFs
basis <- create_rbf_object(M=3)
out <- infer_profiles_vb(X = binomial_data, model = "binomial",
   basis = basis, is_parallel = FALSE, vb_max_iter = 10)

#-------------------------------------

basis <- create_rbf_object(M=3)
out <- infer_profiles_vb(X = gaussian_data, model = "gaussian",
   basis = basis, is_parallel = FALSE, vb_max_iter = 10)
```

---

inner_predict_model_expr

(INNER) Predict expression

Description

This functions makes predictions of gene expression levels using a model trained on methylation features extracted from genomic regions.

Usage

```
inner_predict_model_expr(model, test, is_summary = TRUE)
```

Arguments

model	The fitted regression model, i.e. the output of inner_train_model_expr.
test	The testing data.
is_summary	Logical, print the summary statistics.

Value

A list containing the following elements:

- test_pred: The predicted values for the test data.
- test_errors: The test error metrics.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

predict_expr

Examples

```
# Create synthetic data
train_data <- data.frame(x = rnorm(20), y=rnorm(20, 1, 4))
test_data <- data.frame(x = rnorm(40), y=rnorm(20, 1, 3))

# Train the model
train_model <- inner_train_model_expr(formula = y~., model_name="svm",
                                      train = train_data)

# Make predictions
res <- inner_predict_model_expr(model = train_model$model, test = test_data)
```

---

inner_train_model_expr

(INNER) Train expression model from methylation profiles

Description

This function trains a regression model for predicting gene expression levels by taking as input the higher order methylation features extracted from specific genomic regions.

Usage

```
inner_train_model_expr(
  formula = NULL,
  model_name = "lm",
  train,
  is_summary = TRUE
)
```

Arguments

formula	An object of class formula, e.g. see lm function. If NULL, the simple linear model is used.
model_name	A string denoting the regression model. Currently, available models are: "svm", "randomForest", "rlm", "mars", "gp", and "lm".
train	The training data.
is_summary	Logical, print the summary statistics.

Value

A list containing the following elements:

- formula: The formula that was used.
- gex_model: The fitted model.
- train_pred The predicted values for the training data.
- train_errors: The training error metrics.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

predict_expr

Examples

```
# Create synthetic data
train_data <- data.frame(x = rnorm(20), y=rnorm(20, 1, 4))
res <- inner_train_model_expr(formula = y~., train = train_data)

# Using a different model
res <- inner_train_model_expr(model_name = "randomForest",
                              train = train_data)
```

---

meth_data *Synthetic bulk methylation data*

Description

A synthetic dataset containinig 600 entries.

Usage

```
meth_data
```

Format

A list with 600 elements, where each element element is an L x 3 matrix of observations, where:

1st column locations of obseravtions

2nd column total trials

3rd column number of successes

Value

Synthetic bulk methylation data

See Also

gex_data, bernoulli_data, binomial_data, beta_data, gaussian_data

old_boxplot_cluster_gex *(DEPRECATED) Boxplot of clustered expression levels*

Description

(DEPRECATED) boxplot_cluster_gex creates a boxplot of clustered gene expression levels which depend on the clustered methylation profiles. Each colour denotes a different cluster.

Usage

```
old_boxplot_cluster_gex(
  bpr_cluster_obj,
  gex,
  main_lab = "Gene expression levels"
)
```

Arguments

bpr_cluster_obj	The output of the bpr_cluster_wrap function.
gex	The vector of gene expression data for each promoter region.
main_lab	The title of the plot

Value

The figure to be plotted in the device.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

old_plot_cluster_prof, old_plot_fitted_profiles

Examples

```
# Cluster methylation profiles using 4 RBFs
obs <- meth_data
basis <- create_rbf_object(M = 4)
res   <- bpr_cluster_wrap(x = obs, K = 3, em_max_iter = 2, opt_itnmax = 3,
                          init_opt_itnmax = 2, is_parallel = FALSE)

# Create the plot
old_boxplot_cluster_gex(bpr_cluster_obj = res, gex = gex_data)
```

`old_plot_cluster_prof` *(DEPRECATED) Plot of clustered methylation profiles*

Description

(DEPRECATED) `plot_cluster_prof` creates a plot of cluster methylation profiles, where each colour denotes a different cluster.

Usage

```
old_plot_cluster_prof(
  bpr_cluster_obj,
  main_lab = "Clustered methylation profiles"
)
```

Arguments

`bpr_cluster_obj`	The output of the `bpr_cluster_wrap` function.
`main_lab`	The title of the plot

Value

The figure to be plotted in the device.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

`old_plot_fitted_profiles`, `old_boxplot_cluster_gex`

Examples

```
# Cluster methylation profiles using 4 RBFs
obs <- meth_data
basis <- create_rbf_object(M = 4)
res   <- bpr_cluster_wrap(x = obs, K = 3, em_max_iter = 2, opt_itnmax = 3,
                          init_opt_itnmax = 2, is_parallel = FALSE)

# Create the plot
old_plot_cluster_prof(bpr_cluster_obj = res)
```

`old_plot_fitted_profiles` *(DEPRECATED) Plot the fit of methylation profiles across a region*

Description

(DEPRECATED) `plot_fitted_profiles` is a simple function for plotting the methylation data across a give region, together with the fit of the methylation profiles.

Usage

```
old_plot_fitted_profiles(
  region,
  X,
  fit_prof,
  fit_mean = NULL,
  title = "Gene promoter",
  ...
)
```

Arguments

`region`	Promoter region number
`X`	Methylation data observations
`fit_prof`	Fitted profile
`fit_mean`	Fitted mean function
`title`	Title of the plot
`...`	Additional parameters

Value

The figure to be plotted in the device.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

`old_plot_cluster_prof`

Examples

```
# Fit methylation profiles using 3 RBFs
obs <- meth_data
y   <- gex_data
basis <- create_rbf_object(M = 3)
out   <- bpr_predict_wrap(x = obs, y = y, basis = basis,
                          is_parallel = FALSE, opt_itnmax = 5)

# Create the plot
```

```
old_plot_fitted_profiles(region = 16, X = meth_data, fit_prof = out)
```

`partition_bulk_dataset` *Partition bulk methylation dataset to training and test set*

Description

Partition bulk methylation dataset to training and test set

Usage

```
partition_bulk_dataset(dt_obj, cpg_train_prcg = 0.5)
```

Arguments

`dt_obj`	BPRMeth data 'region_object' object
`cpg_train_prcg`	Fraction of CpGs in each genomic region to keep for training set.

Value

The BPRMeth data 'region_object' object with the following changes. The 'met' element will now contain only the 'training' data. An additional element called 'met_test' will store the data that will be used during testing to evaluate the imputation performance. These data will not be seen from BPRMeth during inference.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

`create_region_object`, `read_met`, `impute_bulk_met`

Examples

```
# Partition the synthetic data from BPRMeth package
dt <- partition_bulk_dataset(encode_met)
```

plot_cluster_profiles *Plot clustered methylation profiles across a region*

Description

Function for plotting the clusterd methylation profiles across a given region where each colour denotes a different cluster.

Usage

```
plot_cluster_profiles(
  cluster_obj,
  title = "Clustered profiles",
  x_axis = "genomic region",
  y_axis = "met level",
  x_labels = c("Upstream", "", "Centre", "", "Downstream"),
  ...
)
```

Arguments

cluster_obj	Clustered profiles object, i.e. output from cluster_profiles_vb or cluster_profiles_mle functions.
title	Plot title
x_axis	x axis label
y_axis	x axis label
x_labels	x axis ticks labels
...	Additional parameters

Value

A ggplot2 object.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

plot_infer_profiles, plot_predicted_expr, boxplot_cluster_expr

Examples

```
# Cluster methylation profiles using 3 RBFs
basis <- create_rbf_object(M = 3)
# Perform clustering
cl_obj <- cluster_profiles_vb(X = encode_met$met, K = 3, model = "binomial",
           basis = basis, vb_max_iter = 5)
# Create plot
g <- plot_cluster_profiles(cluster_obj = cl_obj)
```

`plot_infer_profiles` *Plot inferred methylation profiles across a region*

Description

Function for plotting the inferred methylation profiles across a given region, and optionally the mean methylation rate together with the observed methylation data, using `ggplot2`.

Usage

```
plot_infer_profiles(
  region = 1,
  obj_prof,
  obj_mean = NULL,
  obs = NULL,
  title = "Inferred profiles",
  x_axis = "genomic region",
  y_axis = "met level",
  x_labels = c("Upstream", "", "Centre", "", "Downstream"),
  ...
)
```

Arguments

`region`	Genomic region number
`obj_prof`	Inferred profile, i.e. output from `infer_profiles_vb` or `infer_profiles_mle`
`obj_mean`	Inferred mean function, i.e. output from `infer_profiles_vb` or `infer_profiles_mle`
`obs`	Methylation data observations, if a list, will extract the specific `region`, if a matrix will plot directly its observations.
`title`	Plot title
`x_axis`	x axis label
`y_axis`	x axis label
`x_labels`	x axis ticks labels
`...`	Additional parameters

Value

A ggplot2 object.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

`plot_predicted_expr`, `plot_cluster_profiles`, `boxplot_cluster_expr`

Examples

```
# Fit methylation profiles using 5 RBFs
basis <- create_rbf_object(M = 5)
prof <- infer_profiles_vb(X = encode_met$met, model = "binomial",
    basis = basis, is_parallel = FALSE, vb_max_iter = 5)
# Create the plot
g <- plot_infer_profiles(region = 16, obj_prof = prof, obs = encode_met$met)
```

`plot_predicted_expr` *Scatter plot of predicted vs measured gene expression levels*

Description

`plot_predicted_expr` creates a scatter plot of predicted gene expression values on the x-axis versus the measured gene expression values on the y-axis.

Usage

```
plot_predicted_expr(
  pred_obj,
  title = "Predicted expression",
  is_margins = FALSE
)
```

Arguments

`pred_obj`	The output of the `predict_expr` function.
`title`	The title of the plot.
`is_margins`	Use specified margins or not.

Value

A ggplot2 object.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

`plot_infer_profiles`, `plot_cluster_profiles`, `boxplot_cluster_expr`

Examples

```
# Fit methylation profiles using 5 RBFs
basis <- create_rbf_object(M = 5)
prof <- infer_profiles_vb(X = encode_met$met, model = "binomial",
    basis = basis, is_parallel = FALSE, vb_max_iter = 5)
# Predict expression
pred_obj <- predict_expr(prof_obj = prof, expr = encode_expr,
  anno = encode_met$anno, model_name = "lm", is_summary = FALSE)
```

```
# Create plot
g <- plot_predicted_expr(pred_obj = pred_obj)
```

`pool_bs_seq_rep` *(DEPRECATED) Read and pool replicates from BS-Seq data*

Description

(DEPRECATED) `pool_bs_seq_rep` reads and pools replicate methylation data from BS-Seq experiments that are either in Encode RRBS or Bismark Cov format. Read the Important section below on when to use this function.

Usage

```
pool_bs_seq_rep(files, file_format = "encode_rrbs", chr_discarded = NULL)
```

Arguments

`files`	A vector of filenames containing replicate experiments. This can also be just a single replicate.
`file_format`	A string denoting the file format that the BS-Seq data are stored. Current version allows "`encode_rrbs`" or "`bismark_cov`" formats.
`chr_discarded`	A vector with chromosome names to be discarded.

Value

A `GRanges` object. The GRanges object contains two additional metadata columns:

- `total_reads`: total reads mapped to each genomic location.
- `meth_reads`: methylated reads mapped to each genomic location.

These columns can be accessed as follows: granges_object$total_reads

Important

Unless you want to create a different workflow when processing the BS-Seq data, you should NOT call this function, since this is a helper function. Instead you should call the `preprocess_bs_seq` function.

Information about the file formats can be found in the following links:

Encode RRBS format: http://rohsdb.cmb.usc.edu/GBshape/cgi-bin/hgTables?db=hg19&hgta_group=regulation&hgta_track=wgEncodeHaibMethylRrbs&hgta_table=wgEncodeHaibMethylRrbsBcbreas
hgta_doSchema=describe+table+schema

Bismark Cov format: http://rnbeads.mpi-inf.mpg.de/data/RnBeads.pdf

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

`read_bs_encode_haib`, `preprocess_bs_seq`

Examples

```
# Obtain the path to the file
bs_file1 <- system.file("extdata", "rrbs.bed", package = "BPRMeth")
bs_file2 <- system.file("extdata", "rrbs.bed", package = "BPRMeth")

# Concatenate the files
bs_files <-  c(bs_file1, bs_file2)
# Pool the replicates
pooled_data <- pool_bs_seq_rep(bs_files)
```

---

predict_expr *Predict gene expression from methylation profiles*

Description

bpr_predict_wrap is a function that wraps all the necessary subroutines for performing prediction on gene expression levels. Initially, it optimizes the parameters of the basis functions so as to learn the methylation profiles. Then, uses the learned parameters / coefficients of the basis functions as input features for performing regression in order to predict the corresponding gene expression levels.

Usage

```
predict_expr(
  formula = NULL,
  prof_obj,
  expr,
  anno,
  model_name = "lm",
  train_ind = NULL,
  train_perc = 0.7,
  fit_feature = "RMSE",
  cov_feature = TRUE,
  is_summary = TRUE
)
```

Arguments

formula	An object of class formula, e.g. see lm function. If NULL, the simple linear regression model is used.
prof_obj	Inferred profiles object. This in general will be the output of 'infer_profiles_'(inference_meth) function.
expr	Gene expression data with two columns in data.frame or data.table format. 1st column will have gene IDs and should have column name "id", 2nd column will have expression levels.
anno	Annotation data as a GRanges object.
model_name	A string denoting the regression model. Currently, available models are: "svm", "randomForest", "rlm", "mars", "gp", and "lm".
train_ind	Optional vector containing the indices for the train set.

train_perc	Optional parameter for defining the percentage of the dataset to be used for training set, the remaining will be the test set.
fit_feature	Use additional feature of how well the profile fits the methylation data. Either NULL for ignoring this feature or one of the following: 1) "RMSE" or 2) "NLL" which will be used as input features for predicting expression.
cov_feature	Logical, whether to use coverage as input feature for predictions.
is_summary	Logical, print the summary statistics.

Value

A 'predict_expr' object which consists of the following variables:

- train: The training data.
- test: The test data.
- model: The fitted regression model.
- train_pred The predicted values for the training data.
- test_pred The predicted values for the test data.
- train_errors: The training error metrics.
- test_errors: The test error metrics.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

infer_profiles_mle, infer_profiles_vb, infer_profiles_gibbs, create_basis,

Examples

```
# Fit methylation profiles using 5 RBFs
basis <- create_rbf_object(M = 5)
prof <- infer_profiles_vb(X = encode_met$met, model = "binomial",
    basis = basis, is_parallel = FALSE, vb_max_iter = 5)
# Predict expression
pred_obj <- predict_expr(prof_obj = prof, expr = encode_expr,
  anno = encode_met$anno, model_name = "lm", is_summary = FALSE)
```

---

preprocess_bs_seq *(DEPRECATED) Pre-process BS-Seq data in any given format*

Description

(DEPRECATED) preprocess_bs_seq is a general function for reading and preprocessing BS-Seq data. If a vector of files is given, these are considered as replicates and are pooled together. Finally, noisy reads are discarded.

Usage

```
preprocess_bs_seq(
  files,
  file_format = "encode_rrbs",
  chr_discarded = NULL,
  min_bs_cov = 4,
  max_bs_cov = 1000
)
```

Arguments

files	A vector of filenames containing replicate experiments. This can also be just a single replicate.
file_format	A string denoting the file format that the BS-Seq data are stored. Current version allows "encode_rrbs" or "bismark_cov" formats.
chr_discarded	A vector with chromosome names to be discarded.
min_bs_cov	The minimum number of reads mapping to each CpG site. CpGs with less reads will be considered as noise and will be discarded.
max_bs_cov	The maximum number of reads mapping to each CpG site. CpGs with more reads will be considered as noise and will be discarded.

Value

A GRanges object. The GRanges object contains two additional metadata columns:

- total_reads: total reads mapped to each genomic location.
- meth_reads: methylated reads mapped to each genomic location.

These columns can be accessed as follows: granges_object$total_reads

Additional Info

Information about the file formats can be found in the following links:

Encode RRBS format: http://rohsdb.cmb.usc.edu/GBshape/cgi-bin/hgTables?db=hg19&hgta_group=regulation&hgta_track=wgEncodeHaibMethylRrbs&hgta_table=wgEncodeHaibMethylRrbsBcbreas
hgta_doSchema=describe+table+schema

Bismark Cov format: http://rnbeads.mpi-inf.mpg.de/data/RnBeads.pdf

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

read_bs_encode_haib pool_bs_seq_rep

Examples

```
# Obtain the path to the files
bs_file <- system.file("extdata", "rrbs.bed", package = "BPRMeth")
bs_data <- preprocess_bs_seq(bs_file, file_format = "encode_rrbs")
```

preprocess_final_HTS_data *(DEPRECATED) Pre-process final HTS data for downstream analysis*

Description

(DEPRECATED) `preprocess_final_HTS_data` performs a final filtering and preprocessing on the data for use in downstream analysis. These include, removing noisy gene expression data, removing or not un-expressed genes and log2-transorming of the FPKM values.

Usage

```
preprocess_final_HTS_data(
  methyl_region,
  prom_reg,
  rna_data,
  gene_log2_transf = TRUE,
  gene_outl_thresh = TRUE,
  gex_outlier = 300
)
```

Arguments

`methyl_region`	Methylation region data, which are the output of the "`create_region_object`" function.
`prom_reg`	A `GRanges` object containing corresponding annotated promoter regions for each entry of the `methyl_region` list.
`rna_data`	A `GRanges` object containing corresponding RNA-Seq data for each entry of the `methyl_region` list. This is the output of the "`read_rna_encode_caltech` function.
`gene_log2_transf`	Logical, whether or not to log2 transform the gene expression data.
`gene_outl_thresh`	Logical, whehter or not to remove outlier gene expression data.
`gex_outlier`	Numeric, denoting the threshold above of which the gene expression data (before the log2 transformation) are considered as noise.

Value

An object which contains following information:

- `methyl_region`: The subset of promoter methylation region data after the filtering process.
- `gex`: A vectoring storing only the corresponding gene expression values for each promoter region.
- `rna_data`: The corresponding gene expression data stored as a GRanges object.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

read_rna_encode_caltech process_haib_caltech_wrap

Examples

```
# Obtain the path to the BS file and then read it
bs_file <- system.file("extdata", "rrbs.bed", package = "BPRMeth")
bs_data <- read_bs_encode_haib(bs_file)

# Create promoter regions
rnaseq_file <- system.file("extdata", "rnaseq.bed", package = "BPRMeth")
annot_data <- read_rna_encode_caltech(rnaseq_file)
prom_region <- create_anno_region(annot_data)

# Create methylation regions
methyl_reg <- create_region_object(bs_data, prom_region,
  filter_empty_region = FALSE)

# Keep only covered genomic regions
cov_ind <- which(!is.na(methyl_reg))
methyl_reg <- methyl_reg[cov_ind]
prom_region <- prom_region[cov_ind, ]
annot_data <- annot_data[cov_ind, ]

# Finally preprocess the HTS data
res <- preprocess_final_HTS_data(methyl_reg, prom_region, annot_data)
```

---

process_haib_caltech_wrap

(DEPRECATED) Wrapper for processing ENCODE HAIB and Caltech HTS

Description

(DEPRECATED) process_haib_caltech_wrap is a wrapper method for processing HTS data and returning the methylation promoter regions and the corresponding gene expression data for those promoter regions. Note that the format of BS-Seq data should be in the Encode Haib bed format and for the RNA-Seq data in Encode Caltech bed format.

Usage

```
process_haib_caltech_wrap(
  bs_files,
  rna_files,
  chrom_size_file = NULL,
  chr_discarded = NULL,
  upstream = -7000,
  downstream = 7000,
  min_bs_cov = 4,
  max_bs_cov = 1000,
  cpg_density = 10,
  sd_thresh = 0.1,
```

```
  ignore_strand = TRUE,
  gene_log2_transf = TRUE,
  gene_outl_thresh = TRUE,
  gex_outlier = 300,
  fmin = -1,
  fmax = 1
)
```

Arguments

bs_files	Filename (or vector of filenames if there are replicates) of the BS-Seq '.bed' formatted data to read values from.
rna_files	Filename of the RNA-Seq '.bed' formatted data to read values from. Currently, this version does not support pooling RNA-Seq replicates.
chrom_size_file	Optional filename containing genome chromosome sizes.
chr_discarded	A vector with chromosome names to be discarded.
upstream	Integer defining the length of bp upstream of TSS for creating the promoter region.
downstream	Integer defining the length of bp downstream of TSS for creating the promoter region.
min_bs_cov	The minimum number of reads mapping to each CpG site. CpGs with less reads will be considered as noise and will be discarded.
max_bs_cov	The maximum number of reads mapping to each CpG site. CpGs with more reads will be considered as noise and will be discarded.
cpg_density	Optional integer defining the minimum number of CpGs that have to be in a methylated region. Regions with less than n CpGs are discarded.
sd_thresh	Optional numeric defining the minimum standard deviation of the methylation change in a region. This is used to filter regions with no methylation change.
ignore_strand	Logical, whether or not to ignore strand information.
gene_log2_transf	Logical, whether or not to log2 transform the gene expression data.
gene_outl_thresh	Logical, whehter or not to remove outlier gene expression data.
gex_outlier	Numeric, denoting the threshold above of which the gene expression data (before the log2 transformation) are considered as noise.
fmin	Optional minimum range value for region location scaling. Under this version, this parameter should be left to its default value.
fmax	Optional maximum range value for region location scaling. Under this version, this parameter should be left to its default value.

Value

A processHTS object which contains following information:

- methyl_region: A list containing methylation data, where each entry in the list is an $L_i X 3$ dimensional matrix, where L_i denotes the number of CpGs found in region i. The columns contain the following information:

1. 1st column: Contains the locations of CpGs relative to TSS. Note that the actual locations are scaled to the (fmin, fmax) region.
2. 2nd column: Contains the total reads of each CpG in the corresponding location.
3. 3rd column: Contains the methylated reads each CpG in the corresponding location.

- `gex`: A vector containing the corresponding gene expression levels for each entry of the `methyl_region` list.
- `prom_region`: A `GRanges` object containing corresponding annotated promoter regions for each entry of the `methyl_region` list. The GRanges object has one additional metadata column named `tss`, which stores the TSS of each promoter.
- `rna_data`: A `GRanges` object containing the corresponding RNA-Seq data for each entry of the `methyl_region` list. The GRanges object has three additional metadata columns which are explained in `read_rna_encode_caltech`
- `upstream`: Integer defining the length of bp upstream of TSS.
- `downstream`: Integer defining the length of bp downstream of TSS.
- `cpg_density`: Integer defining the minimum number of CpGs that have to be in a methylated region. Regions with less than `n` CpGs are discarded.
- `sd_thresh`: Numeric defining the minimum standard deviation of the methylation change in a region. This is used to filter regions with no methylation change.
- `fmin`: Minimum range value for region location scaling.
- `fmax`: Maximum range value for region location scaling.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

Examples

```
# Obtain the path to the files
rrbs_file <- system.file("extdata", "rrbs.bed", package = "BPRMeth")
rnaseq_file <- system.file("extdata", "rnaseq.bed", package = "BPRMeth")
proc_data <- process_haib_caltech_wrap(rrbs_file, rnaseq_file)
```

---

read_anno *Read annotation file*

Description

`read_anno` reads a file containing annotation data, which will be used to select genomic regions of interest for downstream analysis. The annotation file format is the following: "chromosome", "start", "end", "strand", "id", "name" (optional). Read the Important section below for more details.

Usage

```
read_anno(
  file,
  chrom_size_file = NULL,
  chr_discarded = NULL,
  is_centre = FALSE,
  is_window = TRUE,
  upstream = -5000,
  downstream = 5000,
  is_anno_region = TRUE,
  delimiter = "\t"
)
```

Arguments

file	File name.
chrom_size_file	Optional file name to read genome chromosome sizes.
chr_discarded	Optional vector with chromosomes to be discarded.
is_centre	Logical, whether 'start' and 'end' locations are pre-centred. If TRUE, the mean of the locations will be chosen as centre. If FALSE, the 'start' will be chosen as the center; e.g. for genes the 'start' denotes the TSS and we use this as centre to obtain K-bp upstream and downstream of TSS.
is_window	Whether to consider a predefined window region around centre. If TRUE, then 'upstream' and 'downstream' parameters are used, otherwise we consider the whole region from start to end location.
upstream	Integer defining the length of bp upstream of 'centre' for creating the genomic region. If is_window = FALSE, this parameter is ignored.
downstream	Integer defining the length of bp downstream of 'centre' for creating the genomic region. If is_window = FALSE, this parameter is ignored.
is_anno_region	Logical, whether or not to create genomic region. It should be set to TRUE, if you want to use the object as input to create_region_object function.
delimiter	Delimiter format the columns are splitted. Default is tab.

Value

A GRanges object.

The GRanges object contains two or three additional metadata columns:

- id: Feature ID, e.g. Ensembl IDs for each gene.
- centre: The central (middle) location of the genomic region. This is required when transforming 'methylation regions' in the (-1, 1) interval, the 'centre' would be at 0.
- name (Optional) the feature name.

These columns can be accessed as follows: granges_obj$id

Important

- When having strand information, and we are in the antisense strand ("-"), the 'start' is automatically switched with the 'end' location.
- By default columns are considered in tab delimited format.
- The "name" feature is optional.
- When there is no "strand" info, this can be set to "*".

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

read_met, create_anno_region, create_region_object

Examples

```
# Obtain the path to files
file <- system.file("extdata", "dummy_anno.bed", package = "BPRMeth")
anno_dt <- read_anno(file, chr_discarded = c("X"))

# Extract feature id
anno_ids <- anno_dt$id
```

---

read_bs_encode_haib *(DEPRECATED) Read ENCODE HAIB bed formatted BS-Seq file*

Description

(DEPRECATED) read_bs_encode_haib reads a file containing methylation data from BS-Seq experiments using the scan function. The BS-Seq file should be in ENCODE HAIB bed format. Read the Important section below on when to use this function.

Usage

```
read_bs_encode_haib(file, chr_discarded = NULL, is_GRanges = TRUE)
```

Arguments

file	The name of the file to read data values from.
chr_discarded	A vector with chromosome names to be discarded.
is_GRanges	Logical: if TRUE a GRanges object is returned, otherwise a data.frame object is returned.

Value

A GRanges object if is_GRanges is TRUE, otherwise a data.table object.

The GRanges object contains two additional metadata columns:

- total_reads: total reads mapped to each genomic location.
- meth_reads: methylated reads mapped to each genomic location.

These columns can be accessed as follows: granges_object$total_reads

Important

Unless you want to create a different workflow when processing the BS-Seq data, you should NOT call this function, since this is a helper function. Instead you should call the preprocess_bs_seq function.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

pool_bs_seq_rep, preprocess_bs_seq

Examples

```
# Obtain the path to the file and then read it
bs_file <- system.file("extdata", "rrbs.bed", package = "BPRMeth")
bs_data <- read_bs_encode_haib(bs_file)
```

---

read_chrom_size *Read genome chromosome sizes file*

Description

read_chrom_size reads a file containing genome chromosome sizes using the fread function.

Usage

```
read_chrom_size(file, delimiter = "\t")
```

Arguments

file	File name
delimiter	Delimiter format the columns are splitted. Default is tab.

Value

A data.table object.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

read_met, read_anno

Examples

```
chr_file <- system.file("extdata", "hg19.chr.sizes", package = "BPRMeth")
chr_data <- read_chrom_size(chr_file)

# Extract the size of chr1
chr_data[1]
```

---

read_expr *Read expression data file*

Description

read_expr reads a file containing expression data. It should contain two columns: "id", "expr" and by default columns are considered in tab delimited format.

Usage

```
read_expr(file, log2_transf = FALSE, delimiter = "\t")
```

Arguments

file	File name
log2_transf	Logical, whether to log2 transform the expression data.
delimiter	Delimiter format the columns are splitted. Default is tab.

Value

A data.table object.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

read_met, read_anno

Examples

```
# Obtain the path to files
file <- system.file("extdata", "dummy_expr.bed", package = "BPRMeth")
expr_dt <- read_expr(file)

# Extract feature id
expr_ids <- expr_dt$id
```

read_met *Read methylation file*

Description

read_met reads a file containing methylation data using the fread function. Since there are different technologies, e.g. array, bulk BS-Seq, scBS-Seq, and still there is no standard file format, different options are available, check the Important section below on the file format for each type you choose. If a file format is not availabe, you need to read the file and create a GRanges object, where the data are ordered by chromosome and genomic location.

Usage

```
read_met(
  file,
  type = "sc_seq",
  strand_info = FALSE,
  chr_discarded = NULL,
  min_bulk_cov = 4,
  max_bulk_cov = 1000,
  delimiter = "\t"
)
```

Arguments

file	File name.
type	Type of technology as character. Either "bulk_seq", "sc_seq" or "array". Check the Important section below for more details.
strand_info	Logical, whether or not the file contains strand information.
chr_discarded	Optional vector with chromosomes to be discarded.
min_bulk_cov	Minimum number of reads mapping to each CpG site. Used only for "bulk_seq" and CpGs with less reads will be discarded as noise.
max_bulk_cov	Maximum number of reads mapping to each CpG site. Used only for "bulk_seq" and CpGs with less reads will be discarded as noise.
delimiter	Delimiter format the columns are splitted. Default is tab.

Value

A GRanges object.

The GRanges object contains one or two additional metadata columns:

- met: Methylation level.
  - For "array" this is the Beta or M-values
  - For "sc_seq" this is either 0 or 1 (unmethylated or methylated)
  - For "bulk_seq" this contains the number of methylated reads for each CpG.
- total: Total number of reads for each CpG. Present only for "bulk_seq" type.

These columns can be accessed as follows: granges_obj$met

Important

Depending on technology `type` we assume different file formats.

- `"array"` File format: "chromosome", "start", "strand" (optional), "met" .
- `"sc_seq"` File format: "chromosome", "start", "strand" (optional), "met" . Where "met" should contain only 1s or 0s.
- `"bulk_seq"` File format: "chromosome", "start", "strand" (optional), "met", "total".

By default columns are considered in tab delimited format.

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

`read_anno`, `read_expr`, `create_region_object`

Examples

```
# Obtain the path to files
file <- system.file("extdata", "dummy_met.bed", package = "BPRMeth")
met_dt <- read_met(file)

# Extract methylation level
met <- met_dt$met
```

---

read_rna_encode_caltech

(DEPRECATED) Read ENCODE Caltech bed formatted RNA-Seq file

Description

(DEPRECATED) `read_rna_encode_caltech` reads a file containing promoter annotation data together with gene expression levels from RNA-Seq experiments using the `scan` function. The RNA-Seq file should be in ENCODE Caltech `bed` format, e.g. use `gtf2bed` tool if your initial file is in `gtf` format.

Usage

```
read_rna_encode_caltech(file, chr_discarded = NULL, is_GRanges = TRUE)
```

Arguments

`file`	The name of the file to read data values from.
`chr_discarded`	A vector with chromosome names to be discarded.
`is_GRanges`	Logical: if TRUE a GRanges object is returned, otherwise a data.frame object is returned.

Value

A GRanges object if is_GRanges is TRUE, otherwise a data.table object.

The GRanges object contains three additional metadata columns:

- ensembl_id: Ensembl IDs of each gene promoter.
- gene_name: Gene name.
- gene_fpkm: Expression level in FPKM.

These columns can be accessed as follows: granges_object$ensembl_id

Author(s)

C.A.Kapourani <C.A.Kapourani@ed.ac.uk>

See Also

read_chrom_size, read_bs_encode_haib

Examples

```
# Obtain the path to the file and then read it
rnaseq_file <- system.file("extdata", "rnaseq.bed", package = "BPRMeth")
rna_data <- read_rna_encode_caltech(rnaseq_file)
```

Index