Package ‘CimpleG’

December 3, 2025

Type Package

Title A Method to Identify Single CpG Sites for Classification and Deconvolution

Version 1.0.0

Date 2025-11-21

Maintainer Tiago F.V. Maié <tiagomaie@hotmail.com>

Description DNA methylation signatures are usually based on multivariate approaches that require hundreds of sites for predictions. 'CimpleG' is a method for the detection of small CpG methylation signatures used for cell-type classification and deconvolution. 'CimpleG' is time efficient and performs as well as top performing methods for cell-type classification of blood cells and other somatic cells, while basing its prediction on a single DNA methylation site per cell type (but users can also select more sites if they so wish). Users can train cell type classifiers ('CimpleG' based, and others) and directly apply these in a deconvolution of cell mixes context. Altogether, 'CimpleG' provides a complete computational framework for the delineation of DNAm signatures and cellular deconvolution. For more details see Maié et al. (2023) <doi:10.1186/s13059-023-03000-0>.

License GPL (>= 3)

URL https://github.com/CostaLab/CimpleG,

https://costalab.github.io/CimpleG/

BugReports https://github.com/CostaLab/CimpleG/issues

Depends R (>= 4.1.0)

Imports archive, assertthat, broom, butcher, caret, data.table, devtools, dplyr, forcats, ggExtra, ggplot2, ggrepel, ggsci, grDevices, gtools, magrittr, matrixStats, methods, nnls, OneR, parsnip, patchwork, purrr, recipes, rlang, rsample, scales, stats, tibble, tictoc, tidyr, tidyselect, tsutils, tune, utils, vroom, workflows, yardstick

Suggests Biobase, biomaRt, C50, circlize, EpiDISH, furrr, future, future.apply, GEOquery, ggbeeswarm, ggsignif, glmnet, knitr, minfi, mltools, NMF, nnet, plyr, ranger, RColorBrewer, reshape2, Rfast, rmarkdown, spelling, stringr, SummarizedExperiment, testthat (>= 3.0.0), withr, xgboost

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Tiago F.V. Maié [aut, cre]

Repository CRAN

Date/Publication 2025-12-03 20:50:17 UTC

Contents

`CimpleG` *Find simple CpG (CimpleG) signatures.*

Description

Train a classification model using (CpGs) as features for the given target data.

Usage

```
CimpleG(
  train_data,
  train_targets = NULL,
  target_columns = NULL,
  test_data = NULL,
  test_targets = NULL,
 method = c("CimpleG", "CimpleG_parab", "brute_force", "logistic_reg", "decision_tree",
    "boost_tree", "mlp", "rand_forest"),
  pred_type = c("both", "hypo", "hyper"),
  engine = c("glmnet", "xgboost", "nnet", "ranger"),
  rank_method = c("ac_rank", "a_rank", "c_rank"),
  k_folds = 10,
  grid_n = 10,
  param_p = 2,
  n_sigs = 1,
  quantile_threshold = 0.005,
  train_only = FALSE,
  split_data = FALSE,
  run_parallel = FALSE,
  deconvolution_reference = TRUE,
  has_annotation = FALSE,
  save_dir = NULL,
  save_format = c("lz4", "gzip", "bzip2", "xz", "nocomp"),
  verbose = 1,
  targets = NULL
)

cimpleg(
  train_data,
  train_targets = NULL,
  target_columns = NULL,
  test_data = NULL,
  test_targets = NULL,
  method = c("CimpleG", "CimpleG_parab", "brute_force", "logistic_reg", "decision_tree",
    "boost_tree", "mlp", "rand_forest"),
  pred_type = c("both", "hypo", "hyper"),
  engine = c("glmnet", "xgboost", "nnet", "ranger"),
  rank_method = c("ac_rank", "a_rank", "c_rank"),
  k_folds = 10,
  grid_n = 10,
  param_p = 2,
  n_sigs = 1,
  quantile_threshold = 0.005,
  train_only = FALSE,
  split_data = FALSE,
  run_parallel = FALSE,
  deconvolution_reference = TRUE,
  has_annotation = FALSE,
  save_dir = NULL,
  save_format = c("lz4", "gzip", "bzip2", "xz", "nocomp"),
  verbose = 1,
  targets = NULL
)

cpg(
  train_data,
  train_targets = NULL,
  target_columns = NULL,
  test_data = NULL,
  test_targets = NULL,
  method = c("CimpleG", "CimpleG_parab", "brute_force", "logistic_reg", "decision_tree",
    "boost_tree", "mlp", "rand_forest"),
  pred_type = c("both", "hypo", "hyper"),
  engine = c("glmnet", "xgboost", "nnet", "ranger"),
  rank_method = c("ac_rank", "a_rank", "c_rank"),
  k_folds = 10,
  grid_n = 10,
  param_p = 2,
  n_sigs = 1,
  quantile_threshold = 0.005,
  train_only = FALSE,
  split_data = FALSE,
  run_parallel = FALSE,
  deconvolution_reference = TRUE,
  has_annotation = FALSE,
  save_dir = NULL,
  save_format = c("lz4", "gzip", "bzip2", "xz", "nocomp"),
  verbose = 1,
  targets = NULL
)
```

Arguments

`train_data`	Training dataset. A matrix (s x f) with methylation data (Beta values) that will be used to train/find the predictors. Samples (s) must be in rows while features/CpGs (f) must be in columns.
`train_targets`	A data frame with the training target samples one-hot encoded. A data frame with at least 1 column, with as many rows and in the same order as ‘train_data‘. Target columns need to be one-hot encoded, meaning that, for that column the target samples should be encoded as ‘1‘ while every other sample should be encoded as ‘0‘.
`target_columns`	A string specifying the name of the column in ‘train_targets‘ to be used for training. Can be a character vector if there are several columns in ‘train_targets‘ to be used for training. If this argument is a character vector, CimpleG will search for the best predictors for each target sequentially or in parallel depending on the value of ‘run_parallel‘
`test_data`	Testing dataset. A matrix (s x f) with methylation data (Beta values) that will be used to test the performance of the found predictors. Samples (s) must be in rows while features/CpGs (f) must be in columns. If ‘test_data‘ *OR* ‘test_targets‘ are NULL, CimpleG will generate a stratified test dataset based on ‘train_targets‘ by removing 25 samples from ‘train_data‘ and ‘train_targets‘.
`test_targets`	A data frame with the testing target samples one-hot encoded. A data frame with at least 1 column, with as many rows and in the same order as ‘test_data‘. Target columns need to be one-hot encoded, meaning that, for that column the target samples should be encoded as ‘1‘ while every other sample should be encoded as ‘0‘. If ‘test_data‘ *OR* ‘test_targets‘ are NULL, CimpleG will generate a stratified test dataset based on ‘train_targets‘ by removing 25 samples from ‘train_data‘ and ‘train_targets‘.
`method`	A string specifying the method or type of machine learning model/algorithm to be used for training. These are divided in two main groups. * The simple models (classifiers that use a single feature), ‘CimpleG‘ (default), ‘brute_force‘, ‘CimpleG_unscaled‘ or ‘oner‘; * the complex models (classifiers that use several features), ‘logistic_reg‘, ‘decision_tree‘, ‘boost_tree‘, ‘mlp‘ or ‘rand_forest‘.
`pred_type`	A string specifying the type of predictor/CpG to be searched for during training. Only used for simple models. One of ‘both‘ (default), ‘hypo‘ or ‘hyper‘. If ‘hypo‘, only hypomethylated predictors will be considered. If ‘hyper‘, only hypermethylated predictors will be considered.
`engine`	A string specifying the machine learning engine behind ‘method‘. Only used for complex models. Currently not in use.
`rank_method`	A string specifying the ranking strategy to rank the features during training.
`k_folds`	An integer specifying the number of folds (K) to be used in training for the stratified K-fold cross-validation procedure.
`grid_n`	An integer specifying the number of hyperparameter combinations to train for.
`param_p`	An even number in ‘sigma / (delta^param_p)‘. Tunes how much weight will be given to delta when doing feature selection. Default is `2`.
`n_sigs`	Number of signatures to be saved for classification and used in deconvolution. Default is `1`.

quantile_threshold	A number between 0 and 1. Determines how many features will be kept. Default is 0.005.
train_only	A boolean, if TRUE, CimpleG will only train (find predictors) but not test them against a test dataset.
split_data	A boolean, if ‘TRUE‘, it will subset the train data provided, creating a smaller test set that will be used to test the models after training. This parameter is experimental. Default is ‘FALSE‘.
run_parallel	A boolean, if ‘FALSE‘, the default, it will search for predictors for multiple targets sequentially. If ‘TRUE‘ it will search for predictors for multiple targets at the same time (parallel processing) in order to save in computational time. You need to set up ‘future::plan()‘ before running this function.
deconvolution_reference	A boolean, if ‘TRUE‘, it will create a deconvolution reference matrix based on the training data. This can later be used to perform deconvolution. Default is ‘FALSE‘.
has_annotation	A boolean, if ‘TRUE‘, it will get the CpG annotation from Illumina for the generated signature. Default is ‘FALSE‘.
save_dir	If defined it will save the resulting model to the given directory. Default is NULL.
save_format	Only used if save_dir is not NULL. One of "lz4", "gzip", "bzip2","xz", "nocomp". lz4 is the best option, fast compression and loading times, low space usage.
verbose	How verbose you want CimpleG to be while it is running. At 0, no message is displayed, at 3 every message is displayed. Default is 1.
targets	DEPRECATED use ‘target_columns‘.

Value

A CimpleG object with the results per target class.

Examples

```
library("CimpleG")

# read data
data(train_data)
data(train_targets)
data(test_data)
data(test_targets)

# run CimpleG
cimpleg_result <- CimpleG(
  train_data = train_data,
  train_targets = train_targets,
  test_data = test_data,
  test_targets = test_targets,
  method = "CimpleG",
  target_columns = c("glia","neurons")
)

# check signatures
cimpleg_result$signatures
```

compute_ax — *Feature selection function used in the sigma delta space*

Description

Feature selection function used in the sigma delta space

Usage

```
compute_ax(dm, sv, p)
```

Arguments

`dm`	delta (difference in mean values)
`sv`	sigma (sum of variance values)
`p`	even number, the greater 'p' is the more importance will be given to sigma

Value

numeric value, score used for feature selection

compute_diffmeans_sumvar — *Compute diff mean sum var dataframe*

Description

Compute diff mean sum var dataframe

Usage

```
compute_diffmeans_sumvar(data, target_vector)
```

Arguments

`data`	Matrix with beta values that will be used to compute diffmeans sumvar data frame
`target_vector`	boolean vector defining which samples in data are part of the target class

Value

data.frame with computed difference in means and sum of variances for target comparison (target v others)

`darken` *Helper function to darken down a given color.*

Description

Helper function to darken down a given color.

Usage

```
darken(color, factor = 0.5)
```

Arguments

`color`	Color name or hex code of a color
`factor`	Multiplicative factor by which ‘color‘ will be darkened down

Value

a character value, hex color code of the darkened color provided

---

`deconvolution_barplot` *Stacked barplot of deconvolution results*

Description

Stacked barplot of deconvolution results

Usage

```
deconvolution_barplot(
  deconvoluted_data,
  meta_data,
  sample_id_column,
  true_label_column,
  color_dict = NULL,
  show_x_label = FALSE,
  base_size = 14,
  ...
)
```

Arguments

`deconvoluted_data`	Result from running ‘run_deconvolution‘
`meta_data`	Data.frame containing metadata from deconvoluted samples
`sample_id_column`	Name of the column containing the sample id in the meta_data data.frame
`true_label_column`	Name of the column containing the true labels of the samples in the meta_data data.frame
`color_dict`	Named string featuring colors as values and labels (true labels) as names
`show_x_label`	A boolean, if ‘TRUE‘ the sample labels in the X axis will be shown. Default is ‘FALSE‘.
`base_size`	An integer defining the base size of the text in the plot. Default is ‘14‘.
`...`	Parameters passed to the ggplot2::theme function.

Value

A list with the data and the ggplot2 plot object.

`deconvolution_epidish` *EpiDISH deconvolution*

Description

EpiDISH deconvolution

Usage

```
deconvolution_epidish(
  ref_mat,
  new_data,
  epidish_method = "CBS",
  epidish_nuv = seq(0.1, 1, 0.1),
  epidish_maxit = 10000
)
```

Arguments

`ref_mat`	Reference matrix.
`new_data`	New data matrix.
`epidish_method`	One of ‘CBS‘ (Cibersort), ‘RPC‘ (Robust Partial Correlations), ‘CP‘ (Constrained Projection). Default is ‘CBS‘. See ‘EpiDISH‘ documentation for more information.

`epidish_nuv`	A vector of candidate values used for svm. Only used when epidish_method is set to ‘CBS‘. See ‘EpiDISH‘ documentation for more information.
`epidish_maxit`	Integer with the number of max iterations for IWLS (Iterative Weighted Least Squares). Only used when epidish_method is set to ‘RPC‘.

`deconvolution_nmf` *NMF deconvolution*

Description

NMF deconvolution

Usage

```
deconvolution_nmf(weights_mat, values_mat, ...)
```

Arguments

`weights_mat`	Reference matrix.
`values_mat`	New data matrix.
`...`	Extra parameters to be set NMF options. Most relevant parameters are probably ‘method‘ and ‘beta‘.

`deconvolution_nnls` *NNLS deconvolution*

Description

NNLS deconvolution

Usage

```
deconvolution_nnls(dt, compute_cols, ref_mat)
```

Arguments

`dt`	A data.table with the new data with features/predictions on rows and samples on columns.
`compute_cols`	A character vector with the columns for which the deconvolution algorithm should be ran.
`ref_mat`	The reference matrix as created by CimpleG.

`deconv_pred_obs_plot` *Scatter plots of observed (true) vs predicted values for deconvolution.*

Description

Produces one plot panel per number of methods with predictions. Each plot panel has one plot per cell type.

Usage

```
deconv_pred_obs_plot(
  deconv_df,
  true_values_col,
  predicted_cols,
  sample_id_col,
  group_col,
  axis_lims = list(x = c(0, 1), y = c(0, 1))
)
```

Arguments

`deconv_df`	A data.frame with meta.data, true values and predictions for different methods as columns. Each row should be a prediction for a given sample and a given group/cell type.
`true_values_col`	A string with the name of the column with the true values in ‘deconv_df‘.
`predicted_cols`	A vector of strings with the name of the columns with the predictions for different methods in ‘deconv_df‘.
`sample_id_col`	A string with the name of the column with the sample name or ID in ‘deconv_df‘.
`group_col`	A string with the name of the column containing the cell types or groups in ‘deconv_df‘.
`axis_lims`	A list with two entries, ‘x‘ and ‘y‘, defining the limits of the x and y axis of the plot.

Value

list of ggplot2 objects

`deconv_ranking_plot`	*Boxplot and rankings of deconvolution metrics for deconvolution results.*

Description

Produces data with varied deconvolution performance metrics. Produces one boxplot and one ranking plot with the for each metric.

Usage

```
deconv_ranking_plot(
  deconv_df,
  true_values_col,
  predicted_cols,
  sample_id_col,
  group_col,
  metrics = c("rmse", "r_squared", "adj.r.squared", "AIC"),
  custom_colours = NULL
)
```

Arguments

`deconv_df`	A data.frame with meta.data, true values and predictions for different methods as columns. Each row should be a prediction for a given sample and a given group/cell type.
`true_values_col`	A string with the name of the column with the true values in ‘deconv_df‘.
`predicted_cols`	A vector of strings with the name of the columns with the predictions for different methods in ‘deconv_df‘.
`sample_id_col`	A string with the name of the column with the sample name or ID in ‘deconv_df‘.
`group_col`	A string with the name of the column containing the cell types or groups in ‘deconv_df‘.
`metrics`	A list with two entries, ‘x‘ and ‘y‘, defining the limits of the x and y axis of the plot.
`custom_colours`	A named vector with colours, where the names are the values defined in ‘predicted_cols‘. If ‘NULL‘, default colours will be used.

Value

list object with data and deconvolution performance plots

`diffmeans_sumvariance_plot` *Creates the old version of the difference in means by sum of variances plot*

Description

Represent CpGs in the difference in means, sum of variances space. This plot is often used to select CpGs that would be good classifiers. These CpGs are often located on the bottom left and bottom right of this plot.

Usage

```
diffmeans_sumvariance_plot(
  data,
  xcol = "diff_means",
  ycol = "sum_variance",
  feature_id_col = "id",
  is_feature_selected_col = NULL,
  label_var1 = "Target",
  label_var2 = "Others",
  target_vector = NULL,
  mean_cutoff = NULL,
  var_cutoff = NULL,
  threshold_func = NULL,
  func_factor = NULL,
  feats_to_highlight = NULL,
  cpg_ranking_df = NULL,
  color_all_points = NULL,
  plot_density = TRUE,
  density_type = c("density", "histogram", "boxplot", "violin", "densigram"),
  plot_dir = NULL,
  id_tag = NULL,
  file_tag = NULL,
  custom_mods = FALSE
)
```

Arguments

`data`	Data to create difference in means, sum of variances plot. Either a data.frame with ‘xcol‘,‘ycol‘ and ‘feature_id_col‘ or, if ‘target_vector‘ is not ‘NULL‘ a matrix with beta values from which, given the target, the difference in means between the target and others, and the sum of variances within the target and others will be calculated.
`xcol`	Column with x-axis data
`ycol`	Column with y-axis data

`feature_id_col`	Column with the feature ID
`is_feature_selected_col`	NULL or column with TRUE/FALSE for features which should be highlighted as selected
`label_var1`	Label of the target class
`label_var2`	Label of the other classes
`target_vector`	if not NULL a vector target class assignment, see data
`mean_cutoff`	a numeric draw mean cutoff at given position
`var_cutoff`	a numeric draw variance cutoff at given position
`threshold_func`	specification of the parabola function, see examples
`func_factor`	argument to be passed to the parabola function, see examples
`feats_to_highlight`	features (CpGs) to be highlighted in the plot
`cpg_ranking_df`	data.frame with ranked features (CpGs) to be highlighted in the plot, if present must have the following columns: .id, predType, Rank and DiffAndFoldScaledAUPR
`color_all_points`	color that all non-highlighted points should have, argument defaults to NULL, the default color is black
`plot_density`	A boolean, if TRUE (default) the function will produce density plots on top/side of scatterplot
`density_type`	One of "density", "histogram", "boxplot", "violin" or "densigram". Defines the type of density plot if ‘plot_density = TRUE‘
`plot_dir`	path to directory where to save the plot. If NULL (default), plot will not be saved.
`id_tag`	character string to identify plots, is displayed in the plot and present in the file name
`file_tag`	character string to identify plots, tags only the file name
`custom_mods`	a boolean, if TRUE will add some custom labels to the plot. Default is FALSE

Value

a `ggplot2` object with the dmsv plot.

Examples

```
library("CimpleG")

# read data
data(train_data)
data(train_targets)

# make basic plot
plt <- diffmeans_sumvariance_plot(
  train_data,
  target_vector = train_targets$blood_cells == 1
)
print(plt)

# make plot with parabola, colored and highlighted features
df_dmeansvar <- compute_diffmeans_sumvar(
  train_data,
  target_vector = train_targets$blood_cells==1
)
parab_param <- .7
df_dmeansvar$is_selected <- select_features(
    x = df_dmeansvar$diff_means,
    y = df_dmeansvar$sum_variance,
    a = parab_param
)

plt <- diffmeans_sumvariance_plot(
  data=df_dmeansvar,
  label_var1="Leukocytes",
  color_all_points="red",
  is_feature_selected_col="is_selected",
  feats_to_highlight=c("cg10456121"),
  threshold_func=function(x,a) (a*x)^2,
  func_factor=parab_param
)
print(plt)
```

`dmsv_plot` *Creates the old version of the difference in means by sum of variances plot*

Description

Represent CpGs in the difference in means, sum of variances space. This plot is often used to select CpGs that would be good classifiers. These CpGs are often located on the bottom left and bottom right of this plot.

Usage

```
dmsv_plot(
  dat,
  target_vector = NULL,
  x_var = "diff_means",
  y_var = "sum_variance",
  id_var = "id",
  highlight_var = NULL,
  display_var = NULL,
  label_var1 = "Target",
  label_var2 = "Others",
  point_color = "black",
  subtitle = NULL
)
```

Arguments

`dat`	Data to create dmsv plot (difference in means, sum of variances plot). Either a data.frame with ‘x_var‘,‘y_var‘ and ‘id_var‘ or, if ‘target_vector‘ is not ‘NULL‘ a matrix with beta values from which, given the target, the difference in means between the target and others, and the sum of variances within the target and others will be calculated.
`target_vector`	if not NULL a boolean vector with target class assignment, see data
`x_var`	Name of the column with x-axis data (difference of means).
`y_var`	Name of the column with y-axis data (sum of variances).
`id_var`	Name of the column with the feature/CpG ID.
`highlight_var`	(Optional) Name of the column with the highlighted features. Values in this column should be boolean (`TRUE` for selected, `FALSE` for not selected).
`display_var`	(Optional) Name of the column with the features that should be displayed in the plot as a label. Values in this column should be boolean (`TRUE` for feature that should be displayed, `FALSE` for feature that should not be displayed).
`label_var1`	Label of the target class. Default is `"Target"`.
`label_var2`	Label of the other classes. Default is `"Others"`.
`point_color`	Color of the features/CpGs in the plot. Default is `"black"`. If features are highlighted, non-highlighted features will have a lighter color.
`subtitle`	Subtitle to be displayed in the plot. Default is `NULL`.

Value

a `ggplot2` object with the dmsv plot.

Examples

```
library("CimpleG")

# load CimpleG example data
data(train_data)
data(train_targets)

# make basic plot straight from the data
plt <- dmsv_plot(
  dat = train_data,
  target_vector = train_targets$blood_cells == 1
)
print(plt)

# make plot with highlighted features
# first create a diffmeans sumvar data frame from the data
df_dmeansvar <- compute_diffmeans_sumvar(
  train_data,
  target_vector = train_targets$blood_cells==1
)
# adding a column to this data frame \code{hl_col} with random CpGs
# selected (as TRUE) or not (as FALSE) to be highlighted and displayed.
df_dmeansvar$hl_col <- sample(c(TRUE,FALSE),nrow(df_dmeansvar),replace=TRUE,prob=c(0.1,0.9))
df_dmeansvar$dp_col <- df_dmeansvar$hl_col

plt <- dmsv_plot(
  dat=df_dmeansvar,
  highlight_var="hl_col",
  display_var="dp_col",
  label_var1="Leukocytes",
  point_color="red",
  subtitle="method: CimpleG"
)
print(plt)
```

eval_test_data *Evaluation of produced models on test data*

Description

Evaluation of produced models on test data

Usage

```
eval_test_data(test_data, final_model, method = "oner", verbose = 1)
```

Arguments

`test_data`	Test data.
`final_model`	Model to be tested.
`method`	Method used to train model.
`verbose`	How verbose the logs should be.

Value

a data.frame with the evaluation statistics

get_cpg_annotation *Get CpG annotation from Illumina*

Description

Get CpG annotation from Illumina

Usage

```
get_cpg_annotation(
  cpg_id,
  is_epic = TRUE,
  short_annotation = TRUE,
  silence_warnings = TRUE
)
```

Arguments

`cpg_id`	A character vector with the CpG IDs from Illumina to annotate.
`is_epic`	A boolean, if TRUE, the annotation will be fetched from the EPIC array, otherwise from the 450k array. Default is TRUE.
`short_annotation`	A boolean, if TRUE, only a small number of columns from the full annotation reference will be kept. This leads to an easier to read output. Default is TRUE.
`silence_warnings`	A boolean, if TRUE, warnings produced during the downloading and loading of the data will be silenced. Default is TRUE.

Value

A table with the annotated CpGs in the same order as the provided signatures.

Examples

```
library("CimpleG")

# read data
signatures <- c("cg14501977", "cg24548498")

# Get signature annotation
signature_annotation <- get_cpg_annotation(signatures)

# check signature annotation
signature_annotation
```

`lighten` *Helper function to lighten up a given color.*

Description

Helper function to lighten up a given color.

Usage

```
lighten(color, factor = 0.5)
```

Arguments

`color`	Color name or hex code of a color
`factor`	Multiplicative factor by which ‘color‘ will be lightened up

Value

a character value, hex color code of the lightened color provided

`load_object` *Load an R object saved with CimpleG or an RDS file.*

Description

Load an R object saved with CimpleG or an RDS file.

Usage

```
load_object(file_name)
```

Arguments

`file_name`	File name in the working directory or path to file to be loaded. Files saved with `CimpleG::save_object` and `base::saveRDS` files are supported.

Value

the loaded R object

`make_color_palette` *Make color palette data frame*

Description

Make color palette data frame

Usage

```
make_color_palette(classes)
```

Arguments

`classes`	Vector with classes for which to create a color palette

Value

data.frane with colors defined for each class provided

`make_deconv_pred_obs_data` *Make tidy data for use in deconvolution plots*

Description

Produces data with varied deconvolution performance metrics.

Usage

```
make_deconv_pred_obs_data(
  dat,
  true_values_col,
  predicted_cols,
  sample_id_col,
  group_col
)
```

Arguments

`dat`	data.frame with predictions as columns, each row should be a prediction for a given sample and given group/celltype
`true_values_col`	A string with the name of the column with the true values in ‘dat‘. true values should be between 0 and 1.

`predicted_cols`	A vector of strings with the name of the columns with the predictions for different methods in ‘dat‘. predictions should be between 0 and 1
`sample_id_col`	A string with the name of the column with the sample name or ID in ‘dat‘.
`group_col`	A string with the name of the column containing the cell types or groups in ‘dat‘. group col should be a factor, otherwise the function will make it a factor

Value

tibble with tidied up deconvolution performance data in nested fields

---

`make_deconv_ref_matrix`

Build deconvolution reference matrix

Description

Build deconvolution reference matrix

Usage

```
make_deconv_ref_matrix(cpg_obj, ref_data, ref_data_labels, method = NULL)
```

Arguments

`cpg_obj`	A CimpleG object.
`ref_data`	A matrix with the reference data to be used to build the reference matrix.
`ref_data_labels`	A character vector with the true labels of the samples in the ‘reference_data‘.
`method`	Method used to train models in the CimpleG object. If not provided (NULL), method will be taken from the CimpleG object. Creates the old version of the difference in means by sum of variances plot

Value

A list object containing the deconvolution reference matrix

predict.CimpleG — *Predict outcome from a CimpleG signatures on new data*

Description

Predict outcome from a CimpleG signatures on new data

Usage

```
## S3 method for class 'CimpleG'
predict(object, ..., new_data, class_labels = NULL)
```

Arguments

`object`	CimpleG object.
`...`	Not used at the moment.
`new_data`	Data to be predicted, samples should be in rows and features in columns. Last column of ‘new_data‘ should have the target/class labels coded as 0 or 1.
`class_labels`	Class labels of new data if these are not provided directly with it.

Value

prediction object, list with an entry for each signature

run_deconvolution — *Perform deconvolution on a new set of samples, based on the CimpleG models trained*

Description

Perform deconvolution on a new set of samples, based on the CimpleG models trained

Usage

```
run_deconvolution(
  cpg_obj = NULL,
  new_data = NULL,
  ref_mat = NULL,
  deconvolution_method = c("NNLS", "EpiDISH", "NMF"),
  ...
)
```

Arguments

`cpg_obj`	A CimpleG object. When creating/training CimpleG the parameter 'deconvolution_reference' should be set to 'TRUE'.
`new_data`	Matrix or data.frame that should have the samples you want to perform deconvolution on. Samples should be in rows and probes/CpGs in columns.
`ref_mat`	If the CimpleG object does not have the reference matrix, you can provide it here instead. See 'make_deconv_ref_matrix'
`deconvolution_method`	Deconvolution method to be used. One of #TODO
`...`	Extra parameters only used when deconvolution_method is set to 'NMF'. The most relevant parameter are probably 'method' and 'beta'.

Value

a data.table with the deconvolution results

`save_object` *Save an R object to disk with fast and efficient compression algorithms.*

Description

Save an R object to disk with fast and efficient compression algorithms.

Usage

```
save_object(object, file_name, file_format = "lz4")
```

Arguments

`object`	Object to be saved to disk.
`file_name`	Name of the file where the R object is saved to.
`file_format`	One of "lz4", "gzip", "bzip2","xz", "nocomp". `lz4` is the best option, fast compression and loading times, low space usage. Format "lz4" is only available if package `archive` is installed. Format "zstd" is not supported anymore as the library now needs to be precompiled with R.

Value

NULL invisibly

`select_features` *Feature selection function used in the diffmeans, sumvariance space*

Description

Feature selection function used in the diffmeans, sumvariance space

Usage

```
select_features(x, y, a)
```

Arguments

`x`	difference in means value
`y`	sum of variances value
`a`	parabola parameter, scales how open/closed the parabola is, the higher the value, the more closed the parabola is.

Value

bool vector

`signature_plot` *CpG signature plot*

Description

CpG signature plot

Usage

```
signature_plot(
  cpg_obj,
  data,
  meta_data,
  sample_id_column,
  true_label_column,
  color_dict = NULL,
  color_others = "black",
  as_panel = TRUE,
  is_beta = TRUE,
  base_size = 14,
  ...
)
```

Arguments

`cpg_obj`	A CimpleG object, as generated by the CimpleG function. Alternatively a names character vector or list with the signatures.
`data`	Matrix or data.frame that should have the samples and signatures to plot. Samples should be in rows and probes/CpGs in columns.
`meta_data`	Data.frame containing metadata from samples in 'data'.
`sample_id_column`	Name of the column containing the sample id in the meta_data data.frame
`true_label_column`	Name of the column containing the true labels of the samples in the meta_data data.frame
`color_dict`	Named string featuring colors as values and labels (true labels) as names
`color_others`	The name or hex code of a color by which the non-target samples should be colored by.
`as_panel`	A boolean, if TRUE (default) a single figure panel with all the signatures will be generated. Otherwise, the individual plots will be returned as a list.
`is_beta`	A boolean, if TRUE (default) the values will be plotted in a scale suitable for Beta values. Otherwise, the values will be plotted in scale suitable for M values.
`base_size`	An integer defining the base size of the text in the plot. Default is '14'.
`...`	Parameters passed to the ggplot2::theme function.

Value

A list with the data and the ggplot2 plot object.

`test_data` *Cell line test data*

Description

Cell line test data

Usage

```
test_data
```

Format

A matrix with beta values for 1000 CpGs. Features/variables as columns and 170 samples as rows test_targets *Cell line test data targets*

Description

Cell line test data targets

Usage

```
test_targets
```

Format

A data frame with 18 variables for 170 samples as rows.

`gsm` GSM identifier (GEO accession number) of the sample

`cell_type` the cell type of the respective sample

`adipocytes` one-hot encoded (1 or 0) column defining if a given sample is an adipocyte

`astrocytes` one-hot encoded (1 or 0) column defining if a given sample is an astrocyte

`blood_cells` one-hot encoded (1 or 0) column defining if a given sample is a blood cell

`endothelial_cells` one-hot encoded (1 or 0) column defining if a given sample is an endothelial cell

`epidermal_cells` one-hot encoded (1 or 0) column defining if a given sample is an epidermal cell

`epithelial_cells` one-hot encoded (1 or 0) column defining if a given sample is an epithelial cell

`fibroblasts` one-hot encoded (1 or 0) column defining if a given sample is a fibroblast

`glia` one-hot encoded (1 or 0) column defining if a given sample is a glia cell

`hepatocytes` one-hot encoded (1 or 0) column defining if a given sample is an hepatocyte

`ips_cells` one-hot encoded (1 or 0) column defining if a given sample is an ipsc

`msc` one-hot encoded (1 or 0) column defining if a given sample is an msc

`muscle_cells` one-hot encoded (1 or 0) column defining if a given sample is a muscle cell

`neurons` one-hot encoded (1 or 0) column defining if a given sample is a neuron

`muscle_sc` one-hot encoded (1 or 0) column defining if a given sample is a muscle stem cell

`group_data` to which dataset these data belong to (`train` or `test`)

`description` the cell type of the respective sample, in long form

`train_data` *Cell line train data*

Description

Cell line train data

Usage

```
train_data
```

Format

A matrix with beta values for 1000 CpGs. Features/variables as columns and 409 samples as rows

`train_targets` *Cell line train data targets*

Description

Cell line train data targets

Usage

```
train_targets
```

Format

A data frame with 18 variables for 409 samples as rows.

`gsm` GSM identifier (GEO accession number) of the sample

`cell_type` the cell type of the respective sample

`adipocytes` one-hot encoded (1 or 0) column defining if a given sample is an adipocyte

`astrocytes` one-hot encoded (1 or 0) column defining if a given sample is an astrocyte

`blood_cells` one-hot encoded (1 or 0) column defining if a given sample is a blood cell

`endothelial_cells` one-hot encoded (1 or 0) column defining if a given sample is an endothelial cell

`epidermal_cells` one-hot encoded (1 or 0) column defining if a given sample is an epidermal cell

`epithelial_cells` one-hot encoded (1 or 0) column defining if a given sample is an epithelial cell

`fibroblasts` one-hot encoded (1 or 0) column defining if a given sample is a fibroblast

`glia` one-hot encoded (1 or 0) column defining if a given sample is a glia cell

`hepatocytes` one-hot encoded (1 or 0) column defining if a given sample is an hepatocyte

`ips_cells` one-hot encoded (1 or 0) column defining if a given sample is an ipsc `msc` one-hot encoded (1 or 0) column defining if a given sample is an msc

`muscle_cells` one-hot encoded (1 or 0) column defining if a given sample is a muscle cell

`neurons` one-hot encoded (1 or 0) column defining if a given sample is a neuron

`muscle_sc` one-hot encoded (1 or 0) column defining if a given sample is a muscle stem cell

`group_data` to which dataset these data belong to (`train` or `test`)

`description` the cell type of the respective sample, in long form

Index