Package ‘cola’

January 22, 2025

Type Package

Title A Framework for Consensus Partitioning

Version 2.13.0

Date 2024-02-26

Depends R (>= 4.0.0)

Imports grDevices, graphics, grid, stats, utils, ComplexHeatmap (>= 2.5.4), matrixStats, GetoptLong, circlize (>= 0.4.7), GlobalOptions (>= 0.1.0), clue, parallel, RColorBrewer, cluster, skmeans, png, mclust, crayon, methods, xml2, microbenchmark, httr, knitr (>= 1.4.0), markdown (>= 1.6), digest, impute, brew, Rcpp (>= 0.11.0), BiocGenerics, eulerr, foreach, doParallel, doRNG, irlba

Suggests genefilter, mvtnorm, testthat (>= 0.3), samr, pamr, kohonen, NMF, WGCNA, Rtsne, umap, clusterProfiler, ReactomePA, DOSE, AnnotationDbi, gplots, hu6800.db, BiocManager, data.tree, dendextend, Polychrome, rmarkdown, simplifyEnrichment, cowplot, flexclust, randomForest, e1071

Description Subgroup classification is a basic task in genomic data analysis, especially for gene expression and DNA methylation data analysis. It can also be used to test the agreement to known clinical annotations, or to test whether there exist significant batch effects. The cola package provides a general framework for subgroup classification by consensus partitioning. It has the following features: 1. It modularizes the consensus partitioning processes that various methods can be easily integrated. 2. It provides rich visualizations for interpreting the results. 3. It allows running multiple methods at the same time and provides functionalities to straightforward compare results. 4. It provides a new method to extract features which are more efficient to separate subgroups. 5. It automatically generates detailed reports for the complete analysis. 6. It allows applying consensus partitioning in a hierarchical manner.

URL `https://github.com/jokergoo/cola`, `https://jokergoo.github.io/cola_collection/`

VignetteBuilder knitr

biocViews Clustering, GeneExpression, Classification, Software

License MIT + file LICENSE

LinkingTo Rcpp

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

git_branch devel

git_last_commit c1c8dc8

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-01-22

Author Zuguang Gu [aut, cre] (ORCID: <`https://orcid.org/0000-0002-7395-8709`>)

Maintainer Zuguang Gu <`z.gu@dkfz.de`>

Contents

`adjust_matrix` *Remove rows with low variance and impute missing values*

Description

Remove rows with low variance and impute missing values

Usage

```
adjust_matrix(m, sd_quantile = 0.05, max_na = 0.25, verbose = TRUE)
```

Arguments

`m`	A numeric matrix.
`sd_quantile`	Cutoff of the quantile of standard deviation. Rows with standard deviation less than it are removed.
`max_na`	Maximum NA fraction in each row. Rows with NA fraction larger than it are removed.
`verbose`	Whether to print messages.

Details

The function uses `impute.knn` to impute missing values, then uses `adjust_outlier` to adjust outliers and removes rows with low standard deviations.

Value

A numeric matrix.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
set.seed(123)
m = matrix(rnorm(100), nrow = 10)
m[sample(length(m), 5)] = NA
m[1, ] = 0
m
m2 = adjust_matrix(m)
m2
```

`adjust_outlier` *Adjust outliers*

Description

Adjust outliers

Usage

```
adjust_outlier(x, q = 0.05)
```

Arguments

`x`	A numeric vector.
`q`	Percentile to adjust.

Details

Vaules larger than percentile `1 - q` are adjusted to the `1 - q` percentile and values smaller than percentile `q` are adjusted to the `q` percentile

Value

A numeric vector with same length as the original one.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
set.seed(123)
x = rnorm(40)
x[1] = 100
adjust_outlier(x)
```

`all_leaves-HierarchicalPartition-method`

All leaves in the hierarchy

Description

All leaves in the hierarchy

Usage

```
## S4 method for signature 'HierarchicalPartition'
all_leaves(object, merge_node = merge_node_param())
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.

Value

A vector of node ID.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
all_leaves(golub_cola_rh)
```

`all_nodes-HierarchicalPartition-method`

All nodes in the hierarchy

Description

All nodes in the hierarchy

Usage

```
## S4 method for signature 'HierarchicalPartition'
all_nodes(object, merge_node = merge_node_param())
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.

Value

A vector of node ID.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
all_nodes(golub_cola_rh)
```

`all_partition_methods` *All supported partitioning methods*

Description

All supported partitioning methods

Usage

```
all_partition_methods()
```

Details

New partitioning methods can be registered by `register_partition_methods`.

Value

A vector of supported partitioning methods.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
all_partition_methods()
```

`all_top_value_methods` *All supported top-value methods*

Description

All supported top-value methods

Usage

```
all_top_value_methods()
```

Details

New top-value methods can be registered by `register_top_value_methods`.

Value

A vector of supported top-value methods.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
all_top_value_methods()
```

`aPAC` *Adapted PAC scores*

Description

Adapted PAC scores

Usage

```
aPAC(consensus_mat)
```

Arguments

`consensus_mat`	A consensus matrix.

Details

For the consensus values x, it is transformed to 1 - x if x < 0.5. After the transformation, for any pair of samples in the consensus matrix, If they are always in a same group or always in different groups, the value x is both to 1. Thus, if the consensus matrix shows stable partitions, values x will be all close to 1. Reflected in the CDF of x, the curve is shifted to the right and the area under CDF curve should be very small.

An aPAC value less than 0.05 is considered as the stable partition, which can be thought the proportion of abmiguous partitioning is less than 0.05.

Value

A numeric value.

Examples

```
data(golub_cola)
aPAC(get_consensus(golub_cola[1, 1], k = 2))
aPAC(get_consensus(golub_cola[1, 1], k = 3))
aPAC(get_consensus(golub_cola[1, 1], k = 4))
aPAC(get_consensus(golub_cola[1, 1], k = 5))
aPAC(get_consensus(golub_cola[1, 1], k = 6))
```

`ATC` *Ability to correlate to other rows*

Description

Ability to correlate to other rows

Usage

```
ATC(mat, cor_fun = stats::cor, min_cor = 0, power = 1, k_neighbours = -1, group = NULL, mc.cores = 1, core
```

Arguments

`mat`	A numeric matrix. ATC score is calculated by rows.
`cor_fun`	A function which calculates correlations.
`min_cor`	Cutoff for the minimal absolute correlation.
`power`	Power on the correlation values.
`k_neighbours`	Nearest k neighbours.
`mc.cores`	Number of cores. This argument will be removed in future versions.
`cores`	Number of cores.
`group`	A categorical variable. If it is specified, the correlation is only calculated for the rows in the same group as current row.
`...`	Pass to `cor_fun`.

Details

For a given row in a matrix, the ATC score is the area above the curve of the curmulative density distribution of the absolute correlation to all other rows. Formally, if `F_i(X)` is the cumulative distribution function of `X` where `X` is the absolute correlation for row i with power `power` (i.e. `x = cor^power`), `ATC_i = 1 - \int_{min_cor}^1 F_i(X)`.

By default the ATC scores are calculated by Pearson correlation, to use Spearman correlation, you can register a new top-value method by:

```
register_top_value_methods(
    "ATC_spearman" = function(m) ATC(m, method = "spearman")
)
```

Similarly, to use a robust correlation method, e.g. `bicor` function, you can do like:

```
register_top_value_methods(
    "ATC_bicor" = function(m) ATC(m, cor_fun = WGCNA::bicor)
)
```

If the number of rows execeeds 30000, it internally uses `ATC_approx`.

Value

A vector of numeric values with the same order as rows in the input matrix.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`https://jokergoo.github.io/cola_supplementary/suppl_1_ATC/suppl_1_ATC.html`

Examples

```
set.seed(12345)
nr1 = 100
mat1 = matrix(rnorm(100*nr1), nrow = nr1)

nr2 = 10
require(mvtnorm)
sigma = matrix(0.8, nrow = nr2, ncol = nr2); diag(sigma) = 1
mat2 = t(rmvnorm(100, mean = rep(0, nr2), sigma = sigma))

nr3 = 50
sigma = matrix(0.5, nrow = nr3, ncol = nr3); diag(sigma) = 1
mat3 = t(rmvnorm(100, mean = rep(0, nr3), sigma = sigma))

mat = rbind(mat1, mat2, mat3)
ATC_score = ATC(mat)
plot(ATC_score, pch = 16, col = c(rep(1, nr1), rep(2, nr2), rep(3, nr3)))
```

`ATC_approx` *Ability to correlate to other rows - an approximated method*

Description

Ability to correlate to other rows - an approximated method

Usage

```
ATC_approx(mat, cor_fun = stats::cor, min_cor = 0, power = 1, k_neighbours = -1,
    mc.cores = 1, cores = mc.cores, n_sampling = c(1000, 500),
    group = NULL, ...)
```

Arguments

`mat`	A numeric matrix. ATC score is calculated by rows.
`cor_fun`	A function which calculates correlations on matrix rows.
`min_cor`	Cutoff for the minimal absolute correlation.
`power`	Power on the correlation values.
`k_neighbours`	Nearest k neighbours. Note when this argument is set, there won't be subset sampling for calculating correlations, whihc means, it will calculate correlation to all other rows.
`mc.cores`	Number of cores. This argument will be removed in future versions.
`cores`	Number of cores.
`n_sampling`	When there are too many rows in the matrix, to get the curmulative distribution of how one row correlates other rows, actually we don't need to use all the rows in the matrix, e.g. 1000 rows can already give a very nice estimation.
`group`	A categorical variable. If it is specified, the correlation is only calculated for the rows in the same group as current row.
`...`	Pass to `cor_fun`.

Details

For a matrix with huge number of rows. It is not possible to calculate correlation to all other rows, thus the correlation is only calculated for a randomly sampled subset of othe rows.

With small numbers of rows of the matrix, `ATC` should be used which calculates the "exact" ATC value, but the value of `ATC` and `ATC_approx` should be very similar.

Examples

```
# There is no example
NULL
```

`cola` *A bottle of cola*

Description

A bottle of cola

Usage

```
cola()
```

Details

Simply serve you a bottle of cola.

The ASCII art is from `http://ascii.co.uk/art/coke` .

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
for(i in 1:10) cola()
```

`cola_opt` *Global parameters*

Description

Global parameters

Usage

```
cola_opt(..., RESET = FALSE, READ.ONLY = NULL, LOCAL = FALSE, ADD = FALSE)
```

Arguments

`...`	Arguments for the parameters, see "details" section.
`RESET`	Whether to reset to default values.
`READ.ONLY`	Please ignore.
`LOCAL`	Please ignore.
`ADD`	Please ignore.

Details

There are following global parameters:

`group_diff` Used in `get_signatures,ConsensusPartition-method` to globally control the minimal difference between subgroups.

`fdr_cutoff` Used in `get_signatures,ConsensusPartition-method` to globally control the cutoff of FDR for the differential signature tests.

`color_set_2` Colors for the predicted subgroups.

`help` Whether to print help messages.

`message` Whether to print messages.

Examples

```
cola_opt
cola_opt$group_diff = 0.2  # e.g. for methylation datasets
cola_opt$fdr_cutoff = 0.1  # e.g. for methylation datasets
cola_opt
cola_opt(RESET = TRUE)
```

`cola_report-ConsensusPartition-method`

Make HTML report from the ConsensusPartition object

Description

Make HTML report from the ConsensusPartition object

Usage

```
## S4 method for signature 'ConsensusPartition'
cola_report(object, output_dir = getwd(),
   title = qq("cola Report for Consensus Partitioning (@{object@top_value_method}:@{object@partition_m
    env = parent.frame())
```

Arguments

`object`	A `ConsensusPartition-class` object.
`output_dir`	The output directory where the report is saved.
`title`	Title of the report.
`env`	Where the objects in the report are found, internally used.

Details

It generates report for a specific combination of top-value method and partitioning method.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`cola_report,ConsensusPartitionList-method`

Examples

```
# There is no example
NULL
```

---

cola_report-ConsensusPartitionList-method

Make HTML report from the ConsensusPartitionList object

Description

Make HTML report from the ConsensusPartitionList object

Usage

```
## S4 method for signature 'ConsensusPartitionList'
cola_report(object, output_dir = getwd(), mc.cores = 1, cores = mc.cores,
    title = "cola Report for Consensus Partitioning", env = parent.frame())
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`output_dir`	The output directory where the report is saved.
`mc.cores`	Multiple cores to use. This argument will be removed in future versions.
`cores`	Number of cores, or a `cluster` object returned by `makeCluster`.
`title`	Title of the report.
`env`	Where the objects in the report are found, internally used.

Details

The `ConsensusPartitionList-class` object contains results for all combinations of top-value methods and partitioning methods. This function generates a HTML report which contains all plots and tables for every combination of method.

The report generation may take a while because it generates A LOT of heatmaps.

Examples of reports can be found at `https://jokergoo.github.io/cola_collection/` .

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
if(FALSE) {
# the following code is runnable
data(golub_cola)
cola_report(golub_cola[c("SD", "MAD"), c("hclust", "skmeans")], output_dir = "~/test_cola_cl_report")
}
```

`cola_report-dispatch` *Method dispatch page for cola_report*

Description

Method dispatch page for `cola_report`.

Dispatch

`cola_report` can be dispatched on following classes:

- `cola_report,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `cola_report,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `cola_report,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method

Examples

```
# no example
NULL
```

`cola_report-HierarchicalPartition-method` *Make HTML report from the HierarchicalPartition object*

Description

Make HTML report from the HierarchicalPartition object

Usage

```
## S4 method for signature 'HierarchicalPartition'
cola_report(object, output_dir = getwd(), mc.cores = 1, cores = mc.cores,
    title = qq("cola Report for Hierarchical Partitioning"),
    env = parent.frame())
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`output_dir`	The output directory where the report is put.
`mc.cores`	Multiple cores to use. This argument will be removed in future versions.
`cores`	Number of cores, or a `cluster` object returned by `makeCluster`.
`title`	Title of the report.
`env`	Where the objects in the report are found, internally used.

Details

This function generates a HTML report which contains all plots for all nodes in the partition hierarchy.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
if(FALSE) {
# the following code is runnable
data(golub_cola_rh)
cola_report(golub_cola_rh, output_dir = "~/test_cola_rh_report")
}
```

`cola_rl` *Example ConsensusPartitionList object*

Description

Example ConsensusPartitionList object

Usage

```
data(cola_rl)
```

Details

Following code was used to generate `cola_rl`:

```
  set.seed(123)
  m = cbind(rbind(matrix(rnorm(20*20, mean = 1,   sd = 0.5), nr = 20),
                  matrix(rnorm(20*20, mean = 0,   sd = 0.5), nr = 20),
                  matrix(rnorm(20*20, mean = 0,   sd = 0.5), nr = 20)),
            rbind(matrix(rnorm(20*20, mean = 0,   sd = 0.5), nr = 20),
                  matrix(rnorm(20*20, mean = 1,   sd = 0.5), nr = 20),
                  matrix(rnorm(20*20, mean = 0,   sd = 0.5), nr = 20)),
            rbind(matrix(rnorm(20*20, mean = 0.5, sd = 0.5), nr = 20),
                  matrix(rnorm(20*20, mean = 0.5, sd = 0.5), nr = 20),
                  matrix(rnorm(20*20, mean = 1,   sd = 0.5), nr = 20))
           ) + matrix(rnorm(60*60, sd = 0.5), nr = 60)
  cola_rl = run_all_consensus_partition_methods(data = m, cores = 6)
```

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rl)
cola_rl
```

`collect_classes-ConsensusPartition-method` *Collect subgroups from ConsensusPartition object*

Description

Collect subgroups from ConsensusPartition object

Usage

```
## S4 method for signature 'ConsensusPartition'
collect_classes(object, internal = FALSE,
    show_row_names = FALSE, row_names_gp = gpar(fontsize = 8),
    anno = object@anno, anno_col = object@anno_col)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`internal`	Used internally.
`show_row_names`	Whether to show row names in the heatmap (which is the column name in the original matrix).
`row_names_gp`	Graphics parameters for row names.
`anno`	A data frame of annotations for the original matrix columns. By default it uses the annotations specified in `consensus_partition` or `run_all_consensus_partition_methods`.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If `anno` is a data frame, `anno_col` should be a named list where names correspond to the column names in `anno`.

Details

The percent membership matrix and the subgroup labels for each k are plotted in the heatmaps.

Same row in all heatmaps corresponds to the same column in the original matrix.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
collect_classes(golub_cola["ATC", "skmeans"])
```

`collect_classes-ConsensusPartitionList-method`

Collect classes from ConsensusPartitionList object

Description

Collect classes from ConsensusPartitionList object

Usage

```
## S4 method for signature 'ConsensusPartitionList'
collect_classes(object, k, show_column_names = FALSE,
    column_names_gp = gpar(fontsize = 8),
    anno = get_anno(object), anno_col = get_anno_col(object),
    simplify = FALSE, ...)
```

Arguments

`object`	A `ConsensusPartitionList-class` object returned by `run_all_consensus_partition_methods`.
`k`	Number of subgroups.
`show_column_names`	Whether to show column names in the heatmap (which is the column name in the original matrix).
`column_names_gp`	Graphics parameters for column names.
`anno`	A data frame of annotations for the original matrix columns. By default it uses the annotations specified in `run_all_consensus_partition_methods`.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If `anno` is a data frame, `anno_col` should be a named list where names correspond to the column names in `anno`.
`simplify`	Internally used.
`...`	Pass to `draw,HeatmapList-method`.

Details

There are following panels in the plot:

- a heatmap showing partitions predicted from all methods where the top annotation is the consensus partition summarized from partitions from all methods, weighted by mean silhouette scores in every single method.
- a row barplot annotation showing the mean silhouette scores for different methods.

The row clustering is applied on the dissimilarity matrix calculated by `cl_dissimilarity` with the `comembership` method.

The brightness of the color corresponds to the silhouette scores for the consensus partition in each method.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
collect_classes(golub_cola, k = 3)
```

`collect_classes-dispatch`

Method dispatch page for collect_classes

Description

Method dispatch page for `collect_classes`.

Dispatch

`collect_classes` can be dispatched on following classes:

- `collect_classes,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `collect_classes,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `collect_classes,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

`collect_classes-HierarchicalPartition-method`

Collect classes from HierarchicalPartition object

Description

Collect classes from HierarchicalPartition object

Usage

```
## S4 method for signature 'HierarchicalPartition'
collect_classes(object, merge_node = merge_node_param(),
    show_row_names = FALSE, row_names_gp = gpar(fontsize = 8),
    anno = get_anno(object[1]), anno_col = get_anno_col(object[1]), ...)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.
`show_row_names`	Whether to show the row names.
`row_names_gp`	Graphic parameters for row names.
`anno`	A data frame of annotations for the original matrix columns. By default it uses the annotations specified in `hierarchical_partition`.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If `anno` is a data frame, `anno_col` should be a named list where names correspond to the column names in `anno`.
`...`	Other arguments.

Details

The function plots the hierarchy of the classes.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
collect_classes(golub_cola_rh)
collect_classes(golub_cola_rh, merge_node = merge_node_param(depth = 2))
```

`collect_plots-ConsensusPartition-method`

Collect plots from ConsensusPartition object

Description

Collect plots from ConsensusPartition object

Usage

```
## S4 method for signature 'ConsensusPartition'
collect_plots(object, verbose = TRUE)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`verbose`	Whether print messages.

Details

Plots by `plot_ecdf`, `collect_classes,ConsensusPartition-method`, `consensus_heatmap`, `membership_heatmap` and `get_signatures` are arranged in one single page, for all avaialble k.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`collect_plots,ConsensusPartitionList-method` collects plots for the `ConsensusPartitionList-class` object.

Examples

```
data(golub_cola)
collect_plots(golub_cola["ATC", "skmeans"])
```

`collect_plots-ConsensusPartitionList-method`

Collect plots from ConsensusPartitionList object

Description

Collect plots from ConsensusPartitionList object

Usage

```
## S4 method for signature 'ConsensusPartitionList'
collect_plots(object, k = 2, fun = consensus_heatmap,
    top_value_method = object@top_value_method,
    partition_method = object@partition_method,
    verbose = TRUE, mc.cores = 1, cores = mc.cores, ...)
```

Arguments

`object`	A `ConsensusPartitionList-class` object from `run_all_consensus_partition_methods`.
`k`	Number of subgroups.
`fun`	Function used to generate plots. Valid functions are `consensus_heatmap`, `plot_ecdf`, `membership_heatmap`, `get_signatures` and `dimension_reduction`.
`top_value_method`	A vector of top-value methods.

`partition_method`	A vector of partitioning methods.
`verbose`	Whether to print message.
`mc.cores`	Number of cores. This argument will be removed in figure versions.
`cores`	Number of cores, or a `cluster` object returned by `makeCluster`.
`...`	other Arguments passed to corresponding `fun`.

Details

Plots for all combinations of top-value methods and parittioning methods are arranged in one single page.

This function makes it easy to directly compare results from multiple methods.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`collect_plots,ConsensusPartition-method` collects plots for a single `ConsensusPartition-class` object.

Examples

```
data(golub_cola)
collect_plots(cola_rl, k = 3)
collect_plots(cola_rl, k = 3, fun = membership_heatmap)
collect_plots(cola_rl, k = 3, fun = get_signatures)
```

`collect_plots-dispatch`

Method dispatch page for collect_plots

Description

Method dispatch page for `collect_plots`.

Dispatch

`collect_plots` can be dispatched on following classes:

- `collect_plots,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `collect_plots,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method

Examples

```
# no example
NULL
```

---

collect_stats-ConsensusPartition-method

Draw and compare statistics for a single method

Description

Draw and compare statistics for a single method

Usage

```
## S4 method for signature 'ConsensusPartition'
collect_stats(object, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`...`	Other arguments.

Details

It is identical to `select_partition_number,ConsensusPartition-method`.

Examples

```
# There is no example
NULL
```

---

collect_stats-ConsensusPartitionList-method

Draw and compare statistics for multiple methods

Description

Draw and compare statistics for multiple methods

Usage

```
## S4 method for signature 'ConsensusPartitionList'
collect_stats(object, k, layout_nrow = 2, all_stats = FALSE, ...)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`k`	Number of subgroups.
`layout_nrow`	Number of rows in the layout
`all_stats`	Whether to show all statistics that were calculated. Used internally.
`...`	Other arguments

Details

It draws heatmaps for statistics for multiple methods in parallel, so that users can compare which combination of methods gives the best results with given the number of subgroups.

Examples

```
data(golub_cola)
collect_stats(golub_cola, k = 3)
```

`collect_stats-dispatch`

Method dispatch page for collect_stats

Description

Method dispatch page for `collect_stats`.

Dispatch

`collect_stats` can be dispatched on following classes:

- `collect_stats,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `collect_stats,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

`colnames-ConsensusPartition-method`

Column names of the matrix

Description

Column names of the matrix

Usage

```
## S4 method for signature 'ConsensusPartition'
colnames(x)
```

Arguments

`x`	A `ConsensusPartition-class` object.

Examples

```
# There is no example
NULL
```

`colnames-ConsensusPartitionList-method`

Column names of the matrix

Description

Column names of the matrix

Usage

```
## S4 method for signature 'ConsensusPartitionList'
colnames(x)
```

Arguments

`x`	A `ConsensusPartitionList-class` object.

Examples

```
# There is no example
NULL
```

`colnames-dispatch` *Method dispatch page for colnames*

Description

Method dispatch page for `colnames`.

Dispatch

`colnames` can be dispatched on following classes:

- `colnames,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `colnames,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `colnames,DownSamplingConsensusPartition-method`, `DownSamplingConsensusPartition-class` class method
- `colnames,HierarchicalPartition-method`, `HierarchicalPartition-class` class method

Examples

```
# no example
NULL
```

`colnames-DownSamplingConsensusPartition-method` *Column names of the matrix*

Description

Column names of the matrix

Usage

```
## S4 method for signature 'DownSamplingConsensusPartition'
colnames(x)
```

Arguments

`x`	A `DownSamplingConsensusPartition-class` object.

Examples

```
# There is no example
NULL
```

---

colnames-HierarchicalPartition-method

Column names of the matrix

Description

Column names of the matrix

Usage

```
## S4 method for signature 'HierarchicalPartition'
colnames(x)
```

Arguments

x	A `HierarchicalPartition-class` object.

Examples

```
# There is no example
NULL
```

---

compare_partitions-ConsensusPartition-method

Compare two partitionings

Description

Compare two partitionings

Usage

```
## S4 method for signature 'ConsensusPartition'
compare_partitions(object, object2, output_file, k1 = 2, k2 = 2,
    dimension_reduction_method = "UMAP",
    id_mapping = guess_id_mapping(rownames(object), "org.Hs.eg.db", FALSE),
    row_km1 = ifelse(k1 == 2, 2, 1),
    row_km2 = ifelse(k1 ==2 && k2 == 2, 2, 1),
    row_km3 = ifelse(k2 == 2, 2, 1))
```

Arguments

`object`	A `ConsensusPartition` object.
`object2`	A `ConsensusPartition` object.
`output_file`	The path of the output HTML file. If it is not specified, the report will be opened in the web browser.
`k1`	Number of subgroups in `object`.
`k2`	Number of subgroups in `object2`.
`dimension_reduction_method`	Which dimension reduction method to use.
`id_mapping`	Pass to `functional_enrichment,ConsensusPartition-method`.
`row_km1`	Number of k-means groups, see Details.
`row_km2`	Number of k-means groups, see Details.
`row_km3`	Number of k-means groups, see Details.

Details

The function produces a HTML report which includes comparisons between two partitioning results.

In the report, there are three heatmaps which visualize A) the signature genes specific in the first partition, B) the signature genes both in the two partitionings and C) the signatures genes specific in the second partition. Argument `row_km1`, `row_km2` and `row_km3` control how many k-means groups should be applied on the three heatmaps.

Examples

```
## Not run:
data(golub_cola)
require(hu6800.db)
x = hu6800ENTREZID
mapped_probes = mappedkeys(x)
id_mapping = unlist(as.list(x[mapped_probes]))
compare_partitions(golub_cola["ATC:skmeans"], golub_cola["SD:kmeans"],
    id_mapping = id_mapping)

## End(Not run)
```

`compare_signatures-ConsensusPartition-method`
Compare Signatures from Different k

Description

Compare Signatures from Different k

Usage

```
## S4 method for signature 'ConsensusPartition'
compare_signatures(object, k = object@k, verbose = interactive(), ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups. Value should be a vector.
`verbose`	Whether to print message.
`...`	Other arguments passed to `get_signatures,ConsensusPartition-method`.

Details

It plots an Euler diagram showing the overlap of signatures from different k.

Examples

```
data(golub_cola)
res = golub_cola["ATC", "skmeans"]
compare_signatures(res)
```

`compare_signatures-dispatch`

Method dispatch page for compare_signatures

Description

Method dispatch page for `compare_signatures`.

Dispatch

`compare_signatures` can be dispatched on following classes:

- `compare_signatures,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `compare_signatures,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

`compare_signatures-HierarchicalPartition-method`
Compare Signatures from Different Nodes

Description

Compare Signatures from Different Nodes

Usage

```
## S4 method for signature 'HierarchicalPartition'
compare_signatures(object, merge_node = merge_node_param(),
    method = c("euler", "upset"), upset_max_comb_sets = 20,
    verbose = interactive(), ...)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.
`method`	Method to visualize.
`upset_max_comb_sets`	Maximal number of combination sets to show.
`verbose`	Whether to print message.
`...`	Other arguments passed to `get_signatures,HierarchicalPartition-method`.

Details

It plots an Euler diagram or a UpSet plot showing the overlap of signatures from different nodes. On each node, the number of subgroups is inferred by `suggest_best_k,ConsensusPartition-method`.

Examples

```
data(golub_cola_rh)
compare_signatures(golub_cola_rh)
```

`concordance` *Concordance to the consensus partition*

Description

Concordance to the consensus partition

Usage

```
concordance(membership_each, class)
```

Arguments

`membership_each`	A matrix which contains partitions in every single runs where columns correspond to runs. The object can be get from `get_membership(..., each = TRUE)`.
`class`	Consensus subgroup labels.

Details

Note subgroup labels in `membership_each` should already be adjusted to the consensus labels, i.e. by `relabel_class`.

The concordance score is the mean proportion of samples having the same subgroup labels as the consensus labels among individual partition runs.

Value

A numeric value.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
membership_each = get_membership(golub_cola["SD", "kmeans"], each = TRUE, k = 3)
consensus_classes = get_classes(golub_cola["SD", "kmeans"], k = 3)$class
concordance(membership_each, consensus_classes)
```

`config_ATC` *Adjust parameters for default ATC method*

Description

Adjust parameters for default ATC method

Usage

```
config_ATC(cor_fun = stats::cor, min_cor = 0, power = 1, k_neighbours = -1, group = NULL, cores = 1, ...)
```

Arguments

`cor_fun`	A function that calculates correlations from a matrix (on matrix rows).
`min_cor`	Cutoff for the minimal absolute correlation.
`power`	Power on the correlation values.
`k_neighbours`	Number of the closest neighbours to use.

`group`	A categorical variable.
`cores`	Number of cores.
`...`	Other arguments passed to `ATC`.

Details

This function changes the default parameters for ATC method. All the arguments in this function all pass to `ATC`.

Examples

```
# use Spearman correlation
config_ATC(cor_fun = function(m) stats::cor(m, method = "spearman"))
# use knn
config_ATC(k_neighbours = 100)
```

`ConsensusPartition-class`

The ConsensusPartition class

Description

The ConsensusPartition class

Methods

The `ConsensusPartition-class` has following methods:

- `consensus_partition`**:** constructor method, run consensus partitioning with a specified top-value method and a partitioning method.
- `select_partition_number,ConsensusPartition-method`**:** make a list of plots for selecting optimized number of subgroups.
- `consensus_heatmap,ConsensusPartition-method`**:** make heatmap of the consensus matrix.
- `membership_heatmap,ConsensusPartition-method`**:** make heatmap of the membership for individual partitions.
- `get_signatures,ConsensusPartition-method`**:** get the signature rows and make heatmap.
- `dimension_reduction,ConsensusPartition-method`**:** make dimension reduction plots.
- `collect_plots,ConsensusPartition-method`**:** make heatmaps for consensus matrix and membership matrix with different number of subgroups.
- `collect_classes,ConsensusPartition-method`**:** make heatmap with subgroups with different numbers.
- `get_param,ConsensusPartition-method`**:** get parameters for the consensus clustering.
- `get_matrix,ConsensusPartition-method`**:** get the original matrix.
- `get_consensus,ConsensusPartition-method`**:** get the consensus matrix.

`get_membership,ConsensusPartition-method`: get the membership of partitions generated from random samplings.

`get_stats,ConsensusPartition-method`: get statistics for the consensus partitioning.

`get_classes,ConsensusPartition-method`: get the consensus subgroup labels and other columns.

`suggest_best_k,ConsensusPartition-method`: guess the best number of subgroups.

`test_to_known_factors,ConsensusPartition-method`: test correlation between predicted subgroups and known factors, if available.

`cola_report,ConsensusPartition-method`: generate a HTML report for the whole analysis.

`functional_enrichment,ConsensusPartition-method`: perform functional enrichment analysis on significant genes if rows in the matrix can be corresponded to genes.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`ConsensusPartitionList-class`

The ConsensusPartitionList class

Description

The ConsensusPartitionList class

Details

The object contains results from all combinations of top-value methods and partitioning methods.

Methods

The `ConsensusPartitionList-class` provides following methods:

`run_all_consensus_partition_methods`: constructor method.

`top_rows_overlap,ConsensusPartitionList-method`: plot the overlaps of top rows under different top-value methods.

`top_rows_heatmap,ConsensusPartitionList-method`: plot the heatmap of top rows under different top-value methods.

`get_classes,ConsensusPartitionList-method`: get consensus subgroup labels merged from all methods.

`get_matrix,ConsensusPartition-method`: get the original matrix.

- **`get_stats,ConsensusPartitionList-method`:** get statistics for the partition for a specified k.
- **`get_membership,ConsensusPartitionList-method`:** get consensus membership matrix summarized from all methods.
- **`suggest_best_k,ConsensusPartitionList-method`:** guess the best number of subgroups for all methods.
- **`collect_plots,ConsensusPartitionList-method`:** collect plots from all combinations of top-value methods and partitioning methods with choosing a plotting function.
- **`collect_classes,ConsensusPartitionList-method`:** make a plot which contains predicted subgroups from all combinations of top-value methods and partitioning methods.
- **`test_to_known_factors,ConsensusPartitionList-method`:** test correlation between predicted subgroups and known annotations, if provided.
- **`cola_report,ConsensusPartitionList-method`:** generate a HTML report for the whole analysis.
- **`functional_enrichment,ConsensusPartitionList-method`:** perform functional enrichment analysis on significant genes if rows in the matrix can be corresponded to genes.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

The `ConsensusPartition-class`.

Examples

```
# There is no example
NULL
```

`consensus_heatmap-ConsensusPartition-method`

Heatmap of the consensus matrix

Description

Heatmap of the consensus matrix

Usage

```
## S4 method for signature 'ConsensusPartition'
consensus_heatmap(object, k, internal = FALSE,
    anno = object@anno, anno_col = get_anno_col(object),
   show_row_names = FALSE, show_column_names = FALSE, row_names_gp = gpar(fontsize = 8),
    simplify = FALSE, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups.
`internal`	Used internally.
`anno`	A data frame of annotations for the original matrix columns. By default it uses the annotations specified in `consensus_partition` or `run_all_consensus_partition_methods`.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If `anno` is a data frame, `anno_col` should be a named list where names correspond to the column names in `anno`.
`show_row_names`	Whether plot row names on the consensus heatmap (which are the column names in the original matrix)
`show_column_names`	Whether show column names.
`row_names_gp`	Graphics parameters for row names.
`simplify`	Internally used.
`...`	other arguments.

Details

For row i and column j in the consensus matrix, the value of corresponding x_ij is the probability of sample i and sample j being in a same group from all partitions.

There are following heatmaps from left to right:

- probability of the sample to stay in the corresponding group
- silhouette scores which measure the distance of an item to the second closest subgroups.
- predicted subgroups
- consensus matrix.
- more annotations if provided as `anno`

One thing that is very important to note is that since we already know the consensus subgroups from consensus partition, in the heatmap, only rows or columns within the group is clustered.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`membership_heatmap,ConsensusPartition-method`

Examples

```
data(golub_cola)
consensus_heatmap(golub_cola["ATC", "skmeans"], k = 3)
```

`consensus_partition` *Consensus partition*

Description

Consensus partition

Usage

```
consensus_partition(data,
    top_value_method = "ATC",
    top_n = NULL,
    partition_method = "skmeans",
    max_k = 6,
    k = NULL,
    sample_by = "row",
    p_sampling = 0.8,
    partition_repeat = 50,
    partition_param = list(),
    anno = NULL,
    anno_col = NULL,
    scale_rows = NULL,
    verbose = TRUE,
    mc.cores = 1, cores = mc.cores,
    prefix = "",
    .env = NULL,
    help = cola_opt$help)
```

Arguments

`data`	A numeric matrix where subgroups are found by columns.
`top_value_method`	A single top-value method. Available methods are in `all_top_value_methods`. Use `register_top_value_methods` to add a new top-value method.
`top_n`	Number of rows with top values. The value can be a vector with length > 1. When n > 5000, the function only randomly sample 5000 rows from top n rows. If `top_n` is a vector, paritition will be applied to every values in `top_n` and consensus partition is summarized from all partitions.
`partition_method`	A single partitioning method. Available methods are in `all_partition_methods`. Use `register_partition_methods` to add a new partition method.

`max_k`	Maximal number of subgroups to try. The function will try for `2:max_k` subgroups
`k`	Alternatively, you can specify a vector k.
`sample_by`	Should randomly sample the matrix by rows or by columns?
`p_sampling`	Proportion of the submatrix which contains the top n rows to sample.
`partition_repeat`	Number of repeats for the random sampling.
`partition_param`	Parameters for the partition method which are passed to `...` in a registered partitioning method. See `register_partition_methods` for detail.
`anno`	A data frame with known annotation of samples. The annotations will be plotted in heatmaps and the correlation to predicted subgroups will be tested.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If `anno` is a data frame, `anno_col` should be a named list where names correspond to the column names in `anno`.
`scale_rows`	Whether to scale rows. If it is `TRUE`, scaling method defined in `register_partition_methods` is used.
`verbose`	Whether print messages.
`mc.cores`	Multiple cores to use. This argument will be removed in future versions.
`cores`	Number of cores, or a `cluster` object returned by `makeCluster`.
`prefix`	Internally used.
`.env`	An environment, internally used.
`help`	Whether to print help messages.

Details

The function performs analysis in following steps:

- calculate scores for rows by top-value method,
- for each top_n value, take top n rows,
- randomly sample `p_sampling` rows from the top_n-row matrix and perform partitioning for `partition_repeats` times,
- collect partitions from all individual partitions and summarize a consensus partition.

Value

A `ConsensusPartition-class` object. Simply type object in the interactive R session to see which functions can be applied on it.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`run_all_consensus_partition_methods` runs consensus partitioning with multiple top-value methods and multiple partitioning methods.

Examples

```
set.seed(123)
m = cbind(rbind(matrix(rnorm(20*20, mean = 1,   sd = 0.5), nr = 20),
                matrix(rnorm(20*20, mean = 0,   sd = 0.5), nr = 20),
                matrix(rnorm(20*20, mean = 0,   sd = 0.5), nr = 20)),
          rbind(matrix(rnorm(20*20, mean = 0,   sd = 0.5), nr = 20),
                matrix(rnorm(20*20, mean = 1,   sd = 0.5), nr = 20),
                matrix(rnorm(20*20, mean = 0,   sd = 0.5), nr = 20)),
          rbind(matrix(rnorm(20*20, mean = 0.5, sd = 0.5), nr = 20),
                matrix(rnorm(20*20, mean = 0.5, sd = 0.5), nr = 20),
                matrix(rnorm(20*20, mean = 1,   sd = 0.5), nr = 20))
         ) + matrix(rnorm(60*60, sd = 0.5), nr = 60)
res = consensus_partition(m, partition_repeat = 10, top_n = c(10, 20, 50))
res
```

`consensus_partition_by_down_sampling`

Consensus partitioning only with a subset of columns

Description

Consensus partitioning only with a subset of columns

Usage

```
consensus_partition_by_down_sampling(data,
    top_value_method = "ATC",
    top_n = NULL,
    partition_method = "skmeans",
    max_k = 6, k = NULL,
    subset = min(round(ncol(data)*0.2), 250), pre_select = TRUE,
    verbose = TRUE, prefix = "", anno = NULL, anno_col = NULL,
    predict_method = "centroid",
    dist_method = c("euclidean", "correlation", "cosine"),
    .env = NULL, .predict = TRUE, mc.cores = 1, cores = mc.cores, ...)
```

Arguments

`data`	A numeric matrix where subgroups are found by columns.
`top_value_method`	A single top-value method. Available methods are in `all_top_value_methods`. Use `register_top_value_methods` to add a new top-value method.

`top_n`	Number of rows with top values. The value can be a vector with length > 1. When n > 5000, the function only randomly sample 5000 rows from top n rows. If `top_n` is a vector, paritition will be applied to every values in `top_n` and consensus partition is summarized from all partitions.
`partition_method`	A single partitioning method. Available methods are in `all_partition_methods`. Use `register_partition_methods` to add a new partition method.
`max_k`	Maximal number of subgroups to try. The function will try for `2:max_k` subgroups
`k`	Alternatively, you can specify a vector k.
`subset`	Number of columns to randomly sample, or a vector of selected indices.
`pre_select`	Whether to pre-select by k-means.
`verbose`	Whether to print messages.
`prefix`	Internally used.
`anno`	Annotation data frame.
`anno_col`	Annotation colors.
`predict_method`	Method for predicting class labels. Possible values are "centroid", "svm" and "randomForest".
`dist_method`	Method for predict the class for other columns.
`.env`	An environment, internally used.
`.predict`	Internally used.
`mc.cores`	Number of cores. This argument will be removed in future versions.
`cores`	Number of cores, or a `cluster` object returned by `makeCluster`.
`...`	All pass to `consensus_partition`.

Details

The function performs consensus partitioning only with a small subset of columns and the class of other columns are predicted by `predict_classes,ConsensusPartition-method`.

Examples

```
## Not run:
data(golub_cola)
m = get_matrix(golub_cola)

set.seed(123)
golub_cola_ds = consensus_partition_by_down_sampling(m, subset = 50,
 anno = get_anno(golub_cola), anno_col = get_anno_col(golub_cola),
 top_value_method = "SD", partition_method = "kmeans")

## End(Not run)
```

`correspond_between_rankings`

Correspond between a list of rankings

Description

Correspond between a list of rankings

Usage

```
correspond_between_rankings(lt, top_n = length(lt[[1]]),
    col = cola_opt$color_set_1[1:length(lt)], ...)
```

Arguments

`lt`	A list of scores under different metrics.
`top_n`	Top n elements to show the correspondance.
`col`	A vector of colors for `lt`.
`...`	Pass to `correspond_between_two_rankings`.

Details

It makes plots for every pairwise comparison in `lt`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
require(matrixStats)
mat = matrix(runif(1000), ncol = 10)
x1 = rowSds(mat)
x2 = rowMads(mat)
x3 = rowSds(mat)/rowMeans(mat)
correspond_between_rankings(lt = list(SD = x1, MAD = x2, CV = x3),
    top_n = 20, col = c("red", "blue", "green"))
```

`correspond_between_two_rankings`

Correspond two rankings

Description

Correspond two rankings

Usage

```
correspond_between_two_rankings(x1, x2, name1, name2,
    col1 = 2, col2 = 3, top_n = round(0.25*length(x1)), transparency = 0.9,
    pt_size = unit(1, "mm"), newpage = TRUE, ratio = c(1, 1, 1))
```

Arguments

`x1`	A vector of scores calculated by one metric.
`x2`	A vector of scores calculated by another metric.
`name1`	Name of the first metric.
`name2`	Name of the second metric.
`col1`	Color for the first metric.
`col2`	Color for the second metric.
`top_n`	Top n elements to show the correspondance.
`transparency`	Transparency of the connecting lines.
`pt_size`	Size of the points, must be a `unit` object.
`newpage`	Whether to plot in a new graphic page.
`ratio`	Ratio of width of the left barplot, connection lines and right barplot. The three values will be scaled to a sum of 1.

Details

In `x1` and `x2`, the i^th element in both vectors corresponds to the same object (e.g. same row if they are calculated from a matrix) but with different scores under different metrics.

`x1` and `x2` are sorted in the left panel and right panel respectively. The top n elements under corresponding metric are highlighted by vertical colored lines in both panels. The left and right panels also shown as barplots of the scores in the two metrics. Between the left and right panels, there are lines connecting the same element (e.g. i^th element in `x1` and `x2`) in the two ordered vectors so that you can see how a same element has two different ranks in the two metrics.

Under the plot is a simple Venn diagram showing the overlaps of the top n elements by the two metrics.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`correspond_between_rankings` draws for more than 2 sets of rankings.

Examples

```
require(matrixStats)
mat = matrix(runif(1000), ncol = 10)
x1 = rowSds(mat)
x2 = rowMads(mat)
correspond_between_two_rankings(x1, x2, name1 = "SD", name2 = "MAD", top_n = 20)
```

`david_enrichment` *Perform DAVID enrichment analysis*

Description

Perform DAVID enrichment analysis

Usage

```
david_enrichment(genes, email,
   catalog = c("GOTERM_CC_FAT", "GOTERM_BP_FAT", "GOTERM_MF_FAT", "KEGG_PATHWAY"),
    idtype = "ENSEMBL_GENE_ID", species = "Homo sapiens")
```

Arguments

`genes`	A vector of gene identifiers.
`email`	The email that user registered on DAVID web service (`https://david.ncifcrf.gov/content.jsp?file=WS.html`).
`catalog`	A vector of function catalogs. Valid values should be in `cola:::DAVID_ALL_CATALOGS`.
`idtype`	ID types for the input gene list. Valid values should be in `cola:::DAVID_ALL_ID_TYPES`.
`species`	Full species name if the ID type is not uniquely mapped to one single species.

Details

This function directly sends the HTTP request to DAVID web service (`https://david.ncifcrf.gov/content.jsp?file=WS.html`) and parses the returned XML. The reason of writing this function is I have problems with other R packages doing DAVID analysis (e.g. RDAVIDWebService, `https://bioconductor.org/packages/devel/bioc/html/RDAVIDWebService.html`) because the rJava package RDAVIDWebService depends on can not be installed on my machine.

Users are encouraged to use more advanced gene set enrichment tools such as clusterProfiler (`http://www.bioconductor.org/packages/release/bioc/html/clusterProfiler.html`), or fgsea (`http://www.bioconductor.org/packages/release/bioc/html/fgsea.html`).

If you want to run this function multiple times, please set time intervals between runs.

Value

A data frame with functional enrichment results.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

Now cola has a replacement function `functional_enrichment` to perform enrichment analysis.

Examples

```
# There is no example
NULL
```

---

dim.ConsensusPartition

Dimension of the matrix

Description

Dimension of the matrix

Usage

```
## S3 method for class 'ConsensusPartition'
dim(x)
```

Arguments

`x`	A `ConsensusPartition-class` object.

Examples

```
# There is no example
NULL
```

`dim.ConsensusPartitionList`

Dimension of the matrix

Description

Dimension of the matrix

Usage

```
## S3 method for class 'ConsensusPartitionList'
dim(x)
```

Arguments

`x`	A `ConsensusPartitionList-class` object.

Examples

```
# There is no example
NULL
```

`dim.DownSamplingConsensusPartition`

Dimension of the matrix

Description

Dimension of the matrix

Usage

```
## S3 method for class 'DownSamplingConsensusPartition'
dim(x)
```

Arguments

`x`	A `DownSamplingConsensusPartition-class` object.

Examples

```
# There is no example
NULL
```

---

dim.HierarchicalPartition

Dimension of the matrix

Description

Dimension of the matrix

Usage

```
## S3 method for class 'HierarchicalPartition'
dim(x)
```

Arguments

x	A `HierarchicalPartition-class` object.

Examples

```
# There is no example
NULL
```

---

dimension_reduction-ConsensusPartition-method

Visualize column after dimension reduction

Description

Visualize samples (the matrix columns) after dimension reduction

Usage

```
## S4 method for signature 'ConsensusPartition'
dimension_reduction(object, k, top_n = NULL,
    method = c("PCA", "MDS", "t-SNE", "UMAP"),
    control = list(), color_by = NULL,
    internal = FALSE, nr = 5000,
    silhouette_cutoff = 0.5, remove = FALSE,
    scale_rows = object@scale_rows, verbose = TRUE, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups.
`top_n`	Top n rows to use. By default it uses all rows in the original matrix.
`method`	Which method to reduce the dimension of the data. `MDS` uses `cmdscale`, `PCA` uses `prcomp`. `t-SNE` uses `Rtsne`. `UMAP` uses `umap`.
`color_by`	If annotation table is set, an annotation name can be set here.
`control`	A list of parameters for `Rtsne` or `umap`.
`internal`	Internally used.
`nr`	If number of matrix rows is larger than this value, random `nr` rows are used.
`silhouette_cutoff`	Cutoff of silhouette score. Data points with values less than it will be mapped with cross symbols.
`remove`	Whether to remove columns which have less silhouette scores than the cutoff.
`scale_rows`	Whether to perform scaling on matrix rows.
`verbose`	Whether print messages.
`...`	Pass to `dimension_reduction,matrix-method`.

Value

Locations of the points.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
dimension_reduction(golub_cola["ATC", "skmeans"], k = 3)
```

`dimension_reduction-dispatch`
Method dispatch page for dimension_reduction

Description

Method dispatch page for `dimension_reduction`.

Dispatch

`dimension_reduction` can be dispatched on following classes:

- `dimension_reduction,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `dimension_reduction,DownSamplingConsensusPartition-method`, `DownSamplingConsensusPartition-class` class method
- `dimension_reduction,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `dimension_reduction,matrix-method`, `matrix-class` class method

Examples

```
# no example
NULL
```

---

dimension_reduction-DownSamplingConsensusPartition-method

Visualize column after dimension reduction

Description

Visualize samples (the matrix columns) after dimension reduction

Usage

```
## S4 method for signature 'DownSamplingConsensusPartition'
dimension_reduction(object, k, top_n = NULL,
    method = c("PCA", "MDS", "t-SNE", "UMAP"),
    control = list(), color_by = NULL,
    internal = FALSE, nr = 5000,
    p_cutoff = 0.05, remove = FALSE,
    scale_rows = TRUE, verbose = TRUE, ...)
```

Arguments

`object`	A `DownSamplingConsensusPartition-class` object.
`k`	Number of subgroups.
`top_n`	Top n rows to use. By default it uses all rows in the original matrix.
`method`	Which method to reduce the dimension of the data. `MDS` uses `cmdscale`, `PCA` uses `prcomp`. `t-SNE` uses `Rtsne`. `UMAP` uses `umap`.
`color_by`	If annotation table is set, an annotation name can be set here.
`control`	A list of parameters for `Rtsne` or `umap`.

`internal`	Internally used.
`nr`	If number of matrix rows is larger than this value, random `nr` rows are used.
`p_cutoff`	Cutoff of p-value of class label prediction. Data points with values higher than it will be mapped with cross symbols.
`remove`	Whether to remove columns which have high p-values than the cutoff.
`scale_rows`	Whether to perform scaling on matrix rows.
`verbose`	Whether print messages.
`...`	Other arguments.

Details

This function is basically very similar as `dimension_reduction,ConsensusPartition-method`.

Value

No value is returned.

Examples

```
data(golub_cola_ds)
dimension_reduction(golub_cola_ds, k = 2)
dimension_reduction(golub_cola_ds, k = 3)
```

`dimension_reduction-HierarchicalPartition-method`

Visualize columns after dimension reduction

Description

Visualize columns after dimension reduction

Usage

```
## S4 method for signature 'HierarchicalPartition'
dimension_reduction(object, merge_node = merge_node_param(),
   parent_node, top_n = NULL, top_value_method = object@list[[1]]@top_value_method,
    method = c("PCA", "MDS", "t-SNE", "UMAP"), color_by = NULL,
    scale_rows = object@list[[1]]@scale_rows, verbose = TRUE, ...)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.
`top_n`	Top n rows to use. By default it uses all rows in the original matrix.
`top_value_method`	Which top-value method to use.

`parent_node`	Parent node. If it is set, the function call is identical to `dimension_reduction(object[parent_node])`
`method`	Which method to reduce the dimension of the data. `MDS` uses `cmdscale`, `PCA` uses `prcomp`. `t-SNE` uses `Rtsne`. `UMAP` uses `umap`.
`color_by`	If annotation table is set, an annotation name can be set here.
`scale_rows`	Whether to perform scaling on matrix rows.
`verbose`	Whether print messages.
`...`	Other arguments passed to `dimension_reduction,ConsensusPartition-method`.

Details

The class IDs are extract at `depth`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
dimension_reduction(golub_cola_rh)
```

---

dimension_reduction-matrix-method

Visualize columns after dimension reduction

Description

Visualize columns after dimension reduction

Usage

```
## S4 method for signature 'matrix'
dimension_reduction(object,
    pch = 16, col = "black", cex = 1, main = NULL,
    method = c("PCA", "MDS", "t-SNE", "UMAP"),
    pc = NULL, control = list(),
    scale_rows = FALSE, nr = 5000,
    internal = FALSE, verbose = TRUE)
```

Arguments

`object`	A numeric matrix.
`method`	Which method to reduce the dimension of the data. `MDS` uses `cmdscale`, `PCA` uses `prcomp`. `t-SNE` uses `Rtsne`. `UMAP` uses `umap`.
`pc`	Which two principle components to visualize
`control`	A list of parameters for `Rtsne` or `umap`.
`pch`	Ahape of points.
`col`	Color of points.
`cex`	Aize of points.
`main`	Title of the plot.
`scale_rows`	Whether perform scaling on matrix rows.
`nr`	If number of matrix rows is larger than this value, random `nr` rows are used.
`internal`	Internally used.
`verbose`	Whether print messages.

Value

Locations of the points.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`DownSamplingConsensusPartition-class`

The DownSamplingConsensusPartition class

Description

The DownSamplingConsensusPartition class

Details

The `DownSamplingConsensusPartition` performs consensus partitioning only with a small subset of columns and the class of other columns are predicted by `predict_classes,ConsensusPartition-method`.

The `DownSamplingConsensusPartition-class` is a child class of `ConsensusPartition-class`. It inherits all methods of `ConsensusPartition-class`.

See Also

The constructor function `consensus_partition_by_down_sampling`.

Examples

```
# There is no example
NULL
```

`FCC` *Flatness of the CDF curve*

Description

Flatness of the CDF curve

Usage

```
FCC(consensus_mat, diff = 0.1)
```

Arguments

`consensus_mat`	A consensus matrix.
`diff`	Difference of F(b) - F(a).

Details

For a in [0, 0.5] and for b in [0.5, 1], the flatness measures the flatness of the CDF curve of the consensus matrix. It is calculated as the maximum width that fits F(b) - F(a) <= diff

Value

A numeric value.

Examples

```
data(golub_cola)
FCC(get_consensus(golub_cola[1, 1], k = 2))
FCC(get_consensus(golub_cola[1, 1], k = 3))
FCC(get_consensus(golub_cola[1, 1], k = 4))
FCC(get_consensus(golub_cola[1, 1], k = 5))
FCC(get_consensus(golub_cola[1, 1], k = 6))
```

`find_best_km` *Find a best k for the k-means clustering*

Description

Find a best k for the k-means clustering

Usage

```
find_best_km(mat, max_km = 15)
```

Arguments

`mat`	A matrix where k-means clustering is executed by rows.
`max_km`	Maximal k to try.

Details

The best k is determined by looking for the knee/elbow of the WSS curve (within-cluster sum of square).

Note this function is only for a rough and quick estimation of the best k.

Examples

```
# There is no example
NULL
```

`functional_enrichment-ANY-method` *Perform functional enrichment on signature genes*

Description

Perform functional enrichment on signature genes

Usage

```
## S4 method for signature 'ANY'
functional_enrichment(object,
    id_mapping = guess_id_mapping(object, org_db, verbose),
    org_db = "org.Hs.eg.db", ontology = "BP",
    min_set_size = 10, max_set_size = 1000,
    verbose = TRUE, prefix = "", ...)
```

Arguments

`object`	A vector of gene IDs.
`id_mapping`	If the gene IDs are not Entrez IDs, a named vector should be provided where the names are the gene IDs and values are the correspoinding Entrez IDs. The value can also be a function that converts gene IDs.
`org_db`	Annotation database.
`ontology`	Following ontologies are allowed: `BP`, `CC`, `MF`, `KEGG`, `Reactome`. `MSigDb` with the gmt file set by `gmt_file` argument, or `gmt` for general gmt gene sets.
`min_set_size`	The minimal size of the gene sets.
`max_set_size`	The maximal size of the gene sets.
`verbose`	Whether to print messages.
`prefix`	Used internally.
`...`	Pass to `enrichGO`, `enrichKEGG`, `enricher`, `enrichDO` or `enrichPathway`.

Details

The function enrichment is applied by clusterProfiler, DOSE or ReactomePA packages.

Value

A data frame.

See Also

`http://bioconductor.org/packages/devel/bioc/vignettes/cola/inst/doc/functional_enrichment.html`

Examples

```
# There is no example
NULL
```

`functional_enrichment-ConsensusPartition-method`

Perform functional enrichment on signature genes

Description

Perform functional enrichment on signature genes

Usage

```
## S4 method for signature 'ConsensusPartition'
functional_enrichment(object, gene_fdr_cutoff = cola_opt$fdr_cutoff, k = suggest_best_k(object, help =
   row_km = NULL, id_mapping = guess_id_mapping(rownames(object), org_db, verbose),
    org_db = "org.Hs.eg.db", ontology = "BP",
    min_set_size = 10, max_set_size = 1000,
    verbose = TRUE, prefix = "", ...)
```

Arguments

`object`	a `ConsensusPartition-class` object from `run_all_consensus_partition_methods`.
`gene_fdr_cutoff`	Cutoff of FDR to define significant signature genes.
`k`	Number of subgroups.
`row_km`	Number of row clusterings by k-means to separate the matrix that only contains signatures.
`id_mapping`	If the gene IDs which are row names of the original matrix are not Entrez IDs, a named vector should be provided where the names are the gene IDs in the matrix and values are correspoinding Entrez IDs. The value can also be a function that converts gene IDs.
`org_db`	Annotation database.
`ontology`	See corresponding argumnet in `functional_enrichment,ANY-method`.
`min_set_size`	The minimal size of the gene sets.
`max_set_size`	The maximal size of the gene sets.
`verbose`	Whether to print messages.
`prefix`	Used internally.
`...`	Pass to `functional_enrichment,ANY-method`.

Details

For how to control the parameters of functional enrichment, see help page of `functional_enrichment,ANY-method`.

Value

A list of data frames which correspond to results for the functional ontologies:

See Also

`http://bioconductor.org/packages/devel/bioc/vignettes/cola/inst/doc/functional_enrichment.html`

Examples

```
# There is no example
NULL
```

---

functional_enrichment-ConsensusPartitionList-method

Perform functional enrichment on signature genes

Description

Perform functional enrichment on signature genes

Usage

```
## S4 method for signature 'ConsensusPartitionList'
functional_enrichment(object, gene_fdr_cutoff = cola_opt$fdr_cutoff,
    id_mapping = guess_id_mapping(rownames(object), org_db, FALSE),
    org_db = "org.Hs.eg.db", ontology = "BP",
    min_set_size = 10, max_set_size = 1000, ...)
```

Arguments

`object`	A `ConsensusPartitionList-class` object from `run_all_consensus_partition_methods`.
`gene_fdr_cutoff`	Cutoff of FDR to define significant signature genes.
`id_mapping`	If the gene IDs which are row names of the original matrix are not Entrez IDs, a named vector should be provided where the names are the gene IDs in the matrix and values are correspoinding Entrez IDs. The value can also be a function that converts gene IDs.
`org_db`	Annotation database.
`ontology`	See corresponding argumnet in `functional_enrichment,ANY-method`.
`min_set_size`	The minimal size of the gene sets.
`max_set_size`	The maximal size of the gene sets.
`...`	Pass to `functional_enrichment,ANY-method`.

Details

For each method, the signature genes are extracted based on the best k.

It calls `functional_enrichment,ConsensusPartition-method` on the consensus partitioning results for each method.

For how to control the parameters of functional enrichment, see help page of `functional_enrichment,ANY-method`.

Value

A list where each element in the list corresponds to enrichment results from a single method.

See Also

http://bioconductor.org/packages/devel/bioc/vignettes/cola/inst/doc/functional_enrichment.html

Examples

```
# There is no example
NULL
```

`functional_enrichment-dispatch`

Method dispatch page for functional_enrichment

Description

Method dispatch page for `functional_enrichment`.

Dispatch

`functional_enrichment` can be dispatched on following classes:

- `functional_enrichment,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `functional_enrichment,ANY-method`, `ANY-class` class method
- `functional_enrichment,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `functional_enrichment,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method

Examples

```
# no example
NULL
```

`functional_enrichment-HierarchicalPartition-method`

Perform functional enrichment on signature genes

Description

Perform functional enrichment on signature genes

Usage

```
## S4 method for signature 'HierarchicalPartition'
functional_enrichment(object, merge_node = merge_node_param(),
    gene_fdr_cutoff = cola_opt$fdr_cutoff,
   row_km = NULL, id_mapping = guess_id_mapping(rownames(object), org_db, verbose),
    org_db = "org.Hs.eg.db", ontology = "BP",
    min_set_size = 10, max_set_size = 1000,
    verbose = TRUE, ...)
```

Arguments

`object`	a `HierarchicalPartition-class` object from `hierarchical_partition`.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.
`gene_fdr_cutoff`	Cutoff of FDR to define significant signature genes.
`row_km`	Number of row clusterings by k-means to separate the matrix that only contains signatures.
`id_mapping`	If the gene IDs which are row names of the original matrix are not Entrez IDs, a named vector should be provided where the names are the gene IDs in the matrix and values are correspoinding Entrez IDs. The value can also be a function that converts gene IDs.
`org_db`	Annotation database.
`ontology`	See corresponding argumnet in `functional_enrichment,ANY-method`.
`min_set_size`	The minimal size of the gene sets.
`max_set_size`	The maximal size of the gene sets.
`verbose`	Whether to print messages.
`...`	Pass to `functional_enrichment,ANY-method`.

Details

For how to control the parameters of functional enrichment, see help page of `functional_enrichment,ANY-method`.

Value

A list of data frames which correspond to results for the functional ontologies:

Examples

```
# There is no example
NULL
```

---

get_anno-ConsensusPartition-method
Get annotations

Description

Get annotations

Usage

```
## S4 method for signature 'ConsensusPartition'
get_anno(object)
```

Arguments

`object`	A `ConsensusPartition-class` object.

Value

A data frame if anno was specified in `run_all_consensus_partition_methods` or `consensus_partition`, or else `NULL`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

---

get_anno-ConsensusPartitionList-method
Get annotations

Description

Get annotations

Usage

```
## S4 method for signature 'ConsensusPartitionList'
get_anno(object)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.

Value

A data frame if `anno` was specified in `run_all_consensus_partition_methods`, or else `NULL`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`get_anno-dispatch` *Method dispatch page for get_anno*

Description

Method dispatch page for `get_anno`.

Dispatch

`get_anno` can be dispatched on following classes:

- `get_anno,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `get_anno,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `get_anno,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `get_anno,DownSamplingConsensusPartition-method`, `DownSamplingConsensusPartition-class` class method

Examples

```
# no example
NULL
```

---

get_anno-DownSamplingConsensusPartition-method

Get annotations

Description

Get annotations

Usage

```
## S4 method for signature 'DownSamplingConsensusPartition'
get_anno(object, reduce = FALSE)
```

Arguments

`object`	A `DownSamplingConsensusPartition-class` object.
`reduce`	Used internally.

Value

A data frame if anno was specified in `consensus_partition_by_down_sampling`, or else `NULL`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_ds)
get_anno(golub_cola_ds)
```

---

get_anno-HierarchicalPartition-method

Get annotations

Description

Get annotations

Usage

```
## S4 method for signature 'HierarchicalPartition'
get_anno(object)
```

Arguments

`object`	A `HierarchicalPartition-class` object.

Value

A data frame if `anno` was specified in `hierarchical_partition`, or `NULL`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

---

get_anno_col-ConsensusPartition-method

Get annotation colors

Description

Get annotation colors

Usage

```
## S4 method for signature 'ConsensusPartition'
get_anno_col(object)
```

Arguments

`object`	A `ConsensusPartition-class` object.

Value

A list of color vectors or else `NULL`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`get_anno_col-ConsensusPartitionList-method`

Get annotation colors

Description

Get annotation colors

Usage

```
## S4 method for signature 'ConsensusPartitionList'
get_anno_col(object)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.

Value

A list of color vectors or else `NULL`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`get_anno_col-dispatch` *Method dispatch page for get_anno_col*

Description

Method dispatch page for `get_anno_col`.

Dispatch

`get_anno_col` can be dispatched on following classes:

- `get_anno_col,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `get_anno_col,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `get_anno_col,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

`get_anno_col-HierarchicalPartition-method`

Get annotation colors

Description

Get annotation colors

Usage

```
## S4 method for signature 'HierarchicalPartition'
get_anno_col(object)
```

Arguments

`object`	A `HierarchicalPartition-class` object.

Value

A list of color vectors or `NULL`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`get_children_nodes-HierarchicalPartition-method`

Test whether a node is a leaf node

Description

Test whether a node is a leaf node

Usage

```
## S4 method for signature 'HierarchicalPartition'
get_children_nodes(object, node, merge_node = merge_node_param())
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`node`	A vector of node IDs.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.

Value

A vector of children nodes.

Examples

```
# There is no example
NULL
```

`get_classes-ConsensusPartition-method`

Get subgroup labels

Description

Get subgroup labels

Usage

```
## S4 method for signature 'ConsensusPartition'
get_classes(object, k = object@k)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups.

Value

A data frame with subgroup labels and other columns which are entropy of the percent membership matrix and the silhouette scores which measure the stability of a sample to stay in its group.

If `k` is not specified, it returns a data frame with subgroup labels from all k.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
obj = golub_cola["ATC", "skmeans"]
get_classes(obj, k = 2)
get_classes(obj)
```

`get_classes-ConsensusPartitionList-method`
Get subgroup labels

Description

Get subgroup labels

Usage

```
## S4 method for signature 'ConsensusPartitionList'
get_classes(object, k)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`k`	Number of subgroups.

Details

The subgroup labels are inferred by merging partitions from all methods by weighting the mean silhouette scores in each method.

Value

A data frame with subgroup labels and other columns which are entropy of the percent membership matrix and the silhouette scores which measure the stability of a sample to stay in its group.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
get_classes(golub_cola, k = 2)
```

`get_classes-dispatch` *Method dispatch page for get_classes*

Description

Method dispatch page for `get_classes`.

Dispatch

`get_classes` can be dispatched on following classes:

- `get_classes,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `get_classes,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `get_classes,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `get_classes,DownSamplingConsensusPartition-method`, `DownSamplingConsensusPartition-class` class method

Examples

```
# no example
NULL
```

`get_classes-DownSamplingConsensusPartition-method` *Get subgroup labels*

Description

Get subgroup labels

Usage

```
## S4 method for signature 'DownSamplingConsensusPartition'
get_classes(object, k = object@k, p_cutoff = 0.05, reduce = FALSE)
```

Arguments

`object`	A `DownSamplingConsensusPartition-class` object.
`k`	Number of subgroups.
`p_cutoff`	Cutoff of p-values of class label prediction. It is only used when `k` is a vector.
`reduce`	Used internally.

Value

If `k` is a scalar, it returns a data frame with two columns:

- the class labels
- the p-value for the prediction of class labels.

If `k` is a vector, it returns a data frame of class labels for each `k`. The class label with prediction p-value > `p_cutoff` is set to `NA`.

Examples

```
data(golub_cola_ds)
get_classes(golub_cola_ds, k = 3)
get_classes(golub_cola_ds)
```

---

get_classes-HierarchicalPartition-method

Get class IDs from the HierarchicalPartition object

Description

Get class IDs from the HierarchicalPartition object

Usage

```
## S4 method for signature 'HierarchicalPartition'
get_classes(object, merge_node = merge_node_param())
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.

Value

A data frame of classes IDs. The class IDs are the node IDs where the subgroup sits in the hierarchy.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
get_classes(golub_cola_rh)
```

---

get_consensus-ConsensusPartition-method

Get consensus matrix

Description

Get consensus matrix

Usage

```
## S4 method for signature 'ConsensusPartition'
get_consensus(object, k)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups.

Details

For row i and column j in the consensus matrix, the value of corresponding x_ij is the probability of sample i and sample j being in the same group from all partitions.

Value

A consensus matrix corresponding to the current k.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
obj = golub_cola["ATC", "skmeans"]
get_consensus(obj, k = 2)
```

---

get_matrix-ConsensusPartition-method

Get the original matrix

Description

Get the original matrix

Usage

```
## S4 method for signature 'ConsensusPartition'
get_matrix(object, full = FALSE, include_all_rows = FALSE)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`full`	Whether to extract the complete original matrix.
`include_all_rows`	Internally used.

Value

A numeric matrix.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
obj = golub_cola["ATC", "skmeans"]
get_matrix(obj)
```

---

get_matrix-ConsensusPartitionList-method

Get the original matrix

Description

Get the original matrix

Usage

```
## S4 method for signature 'ConsensusPartitionList'
get_matrix(object)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.

Value

A numeric matrix.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
get_matrix(golub_cola)
```

`get_matrix-dispatch` *Method dispatch page for get_matrix*

Description

Method dispatch page for `get_matrix`.

Dispatch

`get_matrix` can be dispatched on following classes:

- `get_matrix,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `get_matrix,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `get_matrix,DownSamplingConsensusPartition-method`, `DownSamplingConsensusPartition-class` class method
- `get_matrix,HierarchicalPartition-method`, `HierarchicalPartition-class` class method

Examples

```
# no example
NULL
```

---

get_matrix-DownSamplingConsensusPartition-method

Get the original matrix

Description

Get the original matrix

Usage

```
## S4 method for signature 'DownSamplingConsensusPartition'
get_matrix(object, reduce = FALSE)
```

Arguments

`object`	A `DownSamplingConsensusPartition-class` object.
`reduce`	Whether to return the reduced matrix where columns are randomly sampled.

Value

A numeric matrix

Examples

```
# There is no example
NULL
```

---

get_matrix-HierarchicalPartition-method

Get the original matrix

Description

Get the original matrix

Usage

```
## S4 method for signature 'HierarchicalPartition'
get_matrix(object)
```

Arguments

`object`	A `HierarchicalPartition-class` object.

Value

A numeric matrix.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

---

get_membership-ConsensusPartition-method

Get membership matrix

Description

Get membership matrix

Usage

```
## S4 method for signature 'ConsensusPartition'
get_membership(object, k, each = FALSE)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups.
`each`	Whether to return the percentage membership matrix which is summarized from all partitions or the individual membership in every single partition run.

Details

If `each` == `FALSE`, the value in the membership matrix is the probability to be in one subgroup, while if `each` == `TRUE`, the membership matrix contains the subgroup labels for every single partitions which are from randomly sampling from the original matrix.

The percent membership matrix is calculated by `cl_consensus`.

Value

- If `each` == `FALSE`, it returns a membership matrix where rows correspond to the columns from the subgroups.
- If `each` == `TRUE`, it returns a membership matrix where rows correspond to the columns from the original matrix.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`get_membership,ConsensusPartitionList-method` summarizes membership from partitions from all combinations of top-value methods and partitioning methods.

Examples

```
data(golub_cola)
obj = golub_cola["ATC", "skmeans"]
get_membership(obj, k = 2)
get_membership(obj, k = 2, each = TRUE)
```

---

get_membership-ConsensusPartitionList-method

Get membership matrix

Description

Get membership matrix

Usage

```
## S4 method for signature 'ConsensusPartitionList'
get_membership(object, k)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`k`	Number of subgroups.

Details

The membership matrix (the probability of each sample to be in one subgroup, if assuming columns represent samples) is inferred from the consensus partition of every combination of methods, weighted by the mean silhouette score of the partition for each method. So methods which give unstable partitions have lower weights when summarizing membership matrix from all methods.

Value

A membership matrix where rows correspond to the columns in the original matrix.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`get_membership,ConsensusPartition-method` returns membership matrix for a single top-value method and partitioning method.

Examples

```
data(golub_cola)
get_membership(golub_cola, k = 2)
```

`get_membership-dispatch`

Method dispatch page for get_membership

Description

Method dispatch page for `get_membership`.

Dispatch

`get_membership` can be dispatched on following classes:

- `get_membership,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `get_membership,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method

Examples

```
# no example
NULL
```

`get_param-ConsensusPartition-method`

Get parameters

Description

Get parameters

Usage

```
## S4 method for signature 'ConsensusPartition'
get_param(object, k = object@k, unique = TRUE)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups.
`unique`	Whether to apply `unique` to rows of the returned data frame.

Details

It is mainly used internally.

Value

A data frame of parameters corresponding to the current k. In the data frame, each row corresponds to a partition run.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
obj = golub_cola["ATC", "skmeans"]
get_param(obj)
get_param(obj, k = 2)
get_param(obj, unique = FALSE)
```

---

get_signatures-ConsensusPartition-method

Get signature rows

Description

Get signature rows

Usage

```
## S4 method for signature 'ConsensusPartition'
get_signatures(object, k,
   col = if(scale_rows) c("green", "white", "red") else c("blue", "white", "red"),
    silhouette_cutoff = 0.5,
    fdr_cutoff = cola_opt$fdr_cutoff,
    top_signatures = NULL,
    group_diff = cola_opt$group_diff,
    scale_rows = object@scale_rows, .scale_mean = NULL, .scale_sd = NULL,
    row_km = NULL,
   diff_method = c("Ftest", "ttest", "samr", "pamr", "one_vs_others", "uniquely_high_in_one_group"),
    anno = get_anno(object),
```

```
    anno_col = get_anno_col(object),
    internal = FALSE,
    show_row_dend = FALSE,
    show_column_names = FALSE,
    column_names_gp = gpar(fontsize = 8),
    use_raster = TRUE,
    plot = TRUE, verbose = TRUE, seed = 888,
    left_annotation = NULL, right_annotation = NULL,
    simplify = FALSE, prefix = "", enforce = FALSE, hash = NULL, from_hc = FALSE,
    ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups.
`col`	Colors for the main heatmap.
`silhouette_cutoff`	Cutoff for silhouette scores. Samples with values less than it are not used for finding signature rows. For selecting a proper silhouette cutoff, please refer to `https://www.stat.berkeley.edu/~s133/Cluster2a.html#tth_tAb1.`
`fdr_cutoff`	Cutoff for FDR of the difference test between subgroups.
`top_signatures`	Top signatures with most significant fdr. Note since fdr might be same for multiple rows, the final number of signatures might not be exactly the same as the one that has been set.
`group_diff`	Cutoff for the maximal difference between group means.
`scale_rows`	Whether apply row scaling when making the heatmap.
`.scale_mean`	Internally used.
`.scale_sd`	Internally used.
`row_km`	Number of groups for performing k-means clustering on rows. By default it is automatically selected.
`diff_method`	Methods to get rows which are significantly different between subgroups, see 'Details' section.
`anno`	A data frame of annotations for the original matrix columns. By default it uses the annotations specified in `consensus_partition` or `run_all_consensus_partition_methods`.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If anno is a data frame, `anno_col` should be a named list where names correspond to the column names in anno.
`internal`	Used internally.
`show_row_dend`	Whether show row dendrogram.
`show_column_names`	Whether show column names in the heatmap.
`column_names_gp`	Graphics parameters for column names.
`use_raster`	Internally used.

`plot`	Whether to make the plot.
`verbose`	Whether to print messages.
`seed`	Random seed.
`left_annotation`	Annotation put on the left of the heatmap. It should be a `HeatmapAnnotation-class` object. The number of items should be the same as the number of the original matrix rows. The subsetting to the significant rows are automatically performed on the annotation object.
`right_annotation`	Annotation put on the right of the heatmap. Same format as `left_annotation`.
`simplify`	Only used internally.
`prefix`	Only used internally.
`enforce`	The analysis is cached by default, so that the analysis with the same input will be automatically extracted without rerunning them. Set `enforce` to `TRUE` to enforce the funtion to re-perform the analysis.
`hash`	Userd internally.
`from_hc`	Is the `ConsensusPartition-class` object a node of a `HierarchicalPartition` object?
`...`	Other arguments.

Details

Basically the function applies statistical test for the difference in subgroups for every row. There are following methods which test significance of the difference:

ttest First it looks for the subgroup with highest mean value, compare to each of the other subgroups with t-test and take the maximum p-value. Second it looks for the subgroup with lowest mean value, compare to each of the other subgroups again with t-test and take the maximum p-values. Later for these two list of p-values take the minimal p-value as the final p-value.

samr/pamr use SAM (from samr package)/PAM (from pamr package) method to find significantly different rows between subgroups.

Ftest use F-test to find significantly different rows between subgroups.

one_vs_others For each subgroup i in each row, it uses t-test to compare samples in current subgroup to all other samples, denoted as p_i. The p-value for current row is selected as min(p_i).

uniquely_high_in_one_group The signatures are defined as, if they are uniquely up-regulated in subgroup A, then it must fit following criterions: 1. in a two-group t-test of A ~ other_merged_groups, the statistic must be > 0 (high in group A) and p-value must be significant, and 2. for other groups (excluding A), t-test in every pair of groups should not be significant.

`diff_method` can also be a self-defined function. The function needs two arguments which are the matrix for the analysis and the predicted classes. The function should returns a vector of FDR from the difference test.

Value

A data frame with more than two columns:

`which_row`: row index corresponding to the original matrix.

`fdr`: the FDR.

`km`: the k-means groups if `row_km` is set.

other_columns: the mean value (depending rows are scaled or not) in each subgroup.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
res = golub_cola["ATC", "skmeans"]
tb = get_signatures(res, k = 3)
head(tb)
get_signatures(res, k = 3, top_signatures = 100)
```

`get_signatures-dispatch`

Method dispatch page for get_signatures

Description

Method dispatch page for `get_signatures`.

Dispatch

`get_signatures` can be dispatched on following classes:

- `get_signatures,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `get_signatures,DownSamplingConsensusPartition-method`, `DownSamplingConsensusPartition-class` class method
- `get_signatures,HierarchicalPartition-method`, `HierarchicalPartition-class` class method

Examples

```
# no example
NULL
```

---

get_signatures-DownSamplingConsensusPartition-method

Get signature rows

Description

Get signature rows

Usage

```
## S4 method for signature 'DownSamplingConsensusPartition'
get_signatures(object, k,
    p_cutoff = 1, ...)
```

Arguments

`object`	A `DownSamplingConsensusPartition-class` object.
`k`	Number of subgroups.
`p_cutoff`	Cutoff for p-values of class label prediction. Samples with values higher than it are not used for finding signature rows.
`...`	Other arguments passed to `get_signatures,ConsensusPartition-method`.

Details

This function is very similar as `get_signatures,ConsensusPartition-method`.

Examples

```
data(golub_cola_ds)
get_signatures(golub_cola_ds, k = 2)
get_signatures(golub_cola_ds, k = 3)
```

---

get_signatures-HierarchicalPartition-method

Get signatures rows

Description

Get signatures rows

Usage

```
## S4 method for signature 'HierarchicalPartition'
get_signatures(object, merge_node = merge_node_param(),
    group_diff = object@param$group_diff,
    row_km = NULL, diff_method = "Ftest", fdr_cutoff = object@param$fdr_cutoff,
    scale_rows = object[1]@scale_rows,
    anno = get_anno(object),
    anno_col = get_anno_col(object),
    show_column_names = FALSE, column_names_gp = gpar(fontsize = 8),
    verbose = TRUE, plot = TRUE, seed = 888,
    ...)
```

Arguments

`object`	a `HierarchicalPartition-class` object.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.
`group_diff`	Cutoff for the maximal difference between group means.
`row_km`	Number of groups for performing k-means clustering on rows. By default it is automatically selected.
`diff_method`	Methods to get rows which are significantly different between subgroups.
`fdr_cutoff`	Cutoff for FDR of the difference test between subgroups.
`scale_rows`	whether apply row scaling when making the heatmap.
`anno`	a data frame of annotations for the original matrix columns. By default it uses the annotations specified in `hierarchical_partition`.
`anno_col`	a list of colors (color is defined as a named vector) for the annotations. If anno is a data frame, `anno_col` should be a named list where names correspond to the column names in anno.
`show_column_names`	whether show column names in the heatmap.
`column_names_gp`	Graphic parameters for column names.
`verbose`	whether to print messages.
`plot`	whether to make the plot.
`seed`	Random seed.
`...`	other arguments pass to `get_signatures,ConsensusPartition-method`.

Details

The function calls `get_signatures,ConsensusPartition-method` to find signatures at each node of the partition hierarchy.

Value

A data frame with more than two columns:

`which_row`: row index corresponding to the original matrix.

`km`: the k-means groups if `row_km` is set.

other_columns: the mean value (depending rows are scaled or not) in each subgroup.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
tb = get_signatures(golub_cola_rh)
head(tb)
```

`get_stats-ConsensusPartition-method`
Get statistics

Description

Get statistics

Usage

```
## S4 method for signature 'ConsensusPartition'
get_stats(object, k = object@k, all_stats = FALSE)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups. The value can be a vector.
`all_stats`	Whether to show all statistics that were calculated. Used internally.

Details

The statistics are:

1-PAC 1 - proportion of ambiguous clustering, calculated by `PAC`.

mean_silhouette The mean silhouette score. See `https://en.wikipedia.org/wiki/Silhouette_(clustering)` .

concordance The mean probability that each partition fits the consensus partition, calculated by `concordance`.

- **area_increased** The increased area under eCDF (the empirical cumulative distribution function) curve to the previous k.
- **Rand** This is the percent of pairs of samples that are both in a same cluster or both are not in a same cluster in the partition of `k` and `k-1`. See `https://en.wikipedia.org/wiki/Rand_index` .
- **Jaccard** The ratio of pairs of samples are both in a same cluster in the partition of `k` and `k-1` and the pairs of samples are both in a same cluster in the partition `k` or `k-1`.

Value

A matrix of partition statistics.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
obj = golub_cola["ATC", "skmeans"]
get_stats(obj)
get_stats(obj, k = 2)
```

`get_stats-ConsensusPartitionList-method`

Get statistics

Description

Get statistics

Usage

```
## S4 method for signature 'ConsensusPartitionList'
get_stats(object, k, all_stats = FALSE)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`k`	Number of subgroups. The value can only be a single value.
`all_stats`	Whether to show all statistics that were calculated. Used internally.

Value

A matrix of partition statistics for a selected k. Rows in the matrix correspond to combinations of top-value methods and partitioning methods.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
get_stats(golub_cola, k = 2)
```

`get_stats-dispatch` *Method dispatch page for get_stats*

Description

Method dispatch page for `get_stats`.

Dispatch

`get_stats` can be dispatched on following classes:

- `get_stats,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `get_stats,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

`golub_cola` *Example ConsensusPartitionList object from Golub dataset*

Description

Example ConsensusPartitionList object from Golub dataset

Usage

```
data(golub_cola)
```

Details

Following code was used to generate `golub_cola`:

```
library(cola)

library(golubEsets)  # from bioc
data(Golub_Merge)
m = exprs(Golub_Merge)
colnames(m) = paste0("sample_", colnames(m))
anno = pData(Golub_Merge)

m[m <= 1] = NA
m = log10(m)

m = adjust_matrix(m)

library(preprocessCore)  # from bioc
cn = colnames(m)
rn = rownames(m)
m = normalize.quantiles(m)
colnames(m) = cn
rownames(m) = rn

set.seed(123)
golub_cola = run_all_consensus_partition_methods(
    m, cores = 6,
    anno = anno[, c("ALL.AML"), drop = FALSE],
    anno_col = c("ALL" = "red", "AML" = "blue")
)
```

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

https://jokergoo.github.io/cola_examples/Golub_leukemia/

Examples

```
data(golub_cola)
golub_cola
```

`golub_cola_ds` *Example DownSamplingConsensusPartition object from Golub dataset*

Description

Example DownSamplingConsensusPartition object from Golub dataset

Usage

```
data(golub_cola_ds)
```

Details

Following code was used to generate `golub_cola_ds`:

```
    library(cola)
    data(golub_cola)
    m = get_matrix(golub_cola)
    set.seed(123)
    golub_cola_ds = consensus_partition_by_down_sampling(
        m, subset = 50, cores = 6,
        anno = get_anno(golub_cola),
        anno_col = get_anno_col(golub_cola),
    )
```

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_ds)
golub_cola_ds
```

`golub_cola_rh` *Example HierarchicalPartition object from Golub dataset*

Description

Example HierarchicalPartition object from Golub dataset

Usage

```
data(golub_cola_rh)
```

Details

Following code was used to generate `golub_cola_rh`:

```
library(cola)
data(golub_cola)
m = get_matrix(golub_cola)
set.seed(123)
golub_cola_rh = hierarchical_partition(
    m, cores = 6,
    anno = get_anno(golub_cola),
    anno_col = get_anno_col(golub_cola)
)
```

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
golub_cola_rh
```

`HierarchicalPartition-class`

The HierarchicalPartition class

Description

The HierarchicalPartition class

Methods

The `HierarchicalPartition-class` has following methods:

`hierarchical_partition`: constructor method.

`collect_classes,HierarchicalPartition-method`: plot the hierarchy of subgroups predicted.

`get_classes,HierarchicalPartition-method`: get the class IDs of subgroups.

`suggest_best_k,HierarchicalPartition-method`: guess the best number of partitions for each node.

`get_matrix,HierarchicalPartition-method`: get the original matrix.

`get_signatures,HierarchicalPartition-method`: get the signatures for each subgroup.

`compare_signatures,HierarchicalPartition-method`: compare signatures from different nodes.

`dimension_reduction,HierarchicalPartition-method`: make dimension reduction plots.

`test_to_known_factors,HierarchicalPartition-method`: test correlation between predicted subgrouping and known annotations, if available.

`cola_report,HierarchicalPartition-method`: generate a HTML report for the whole analysis.

`functional_enrichment,HierarchicalPartition-method`: apply functional enrichment.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`hierarchical_partition` *Hierarchical partition*

Description

Hierarchical partition

Usage

```
hierarchical_partition(data,
    top_n = NULL,
    top_value_method = "ATC",
    partition_method = "skmeans",
    combination_method =  expand.grid(top_value_method, partition_method),
    anno = NULL, anno_col = NULL,
    mean_silhouette_cutoff = 0.9, min_samples = max(6, round(ncol(data)*0.01)),
    subset = Inf, predict_method = "centroid",
    group_diff = ifelse(scale_rows, 0.5, 0),
    fdr_cutoff = cola_opt$fdr_cutoff,
    min_n_signatures = NULL,
    filter_fun = function(mat) {
    s = rowSds(mat)
    s > quantile(unique(s[s > 1e-10]), 0.05, na.rm = TRUE)
    },
   max_k = 4, scale_rows = TRUE, verbose = TRUE, mc.cores = 1, cores = mc.cores, help = TRUE, ...)
```

Arguments

`data`	a numeric matrix where subgroups are found by columns.
`top_n`	Number of rows with top values.
`top_value_method`	a single or a vector of top-value methods. Available methods are in `all_top_value_methods`.
`partition_method`	a single or a vector of partition methods. Available methods are in `all_partition_methods`.

`combination_method`	A list of combinations of top-value methods and partitioning methods. The value can be a two-column data frame where the first column is the top-value methods and the second column is the partitioning methods. Or it can be a vector of combination names in a form of "top_value_method:partitioning_method".
`anno`	A data frame with known annotation of samples. The annotations will be plotted in heatmaps and the correlation to predicted subgroups will be tested.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If `anno` is a data frame, `anno_col` should be a named list where names correspond to the column names in `anno`.
`mean_silhouette_cutoff`	The cutoff to test whether partition in current node is stable.
`min_samples`	the cutoff of number of samples to determine whether to continue looking for subgroups.
`group_diff`	Pass to `get_signatures,ConsensusPartition-method`.
`fdr_cutoff`	Pass to `get_signatures,ConsensusPartition-method`.
`subset`	Number of columns to randomly sample.
`predict_method`	Method for predicting class labels. Possible values are "centroid", "svm" and "randomForest".
`min_n_signatures`	Minimal number of signatures under the best classification.
`filter_fun`	A self-defined function which filters the original matrix and returns a submatrix for partitioning.
`max_k`	maximal number of partitions to try. The function will try `2:max_k` partitions. Note this is the number of partitions that will be tried out on each node of the hierarchical partition. Since more subgroups will be found in the whole partition hierarchy, on each node, `max_k` should not be set to a large value.
`scale_rows`	Whether rows are scaled?
`verbose`	whether print message.
`mc.cores`	multiple cores to use. This argument will be removed in future versions.
`cores`	Number of cores, or a `cluster` object returned by `makeCluster`.
`help`	Whether to show the help message.
`...`	pass to `consensus_partition`

Details

The function looks for subgroups in a hierarchical way.

There is a special way to encode the node in the hierarchy. The length of the node name is the depth of the node in the hierarchy and the substring excluding the last digit is the name node of the parent node. E.g. for the node `0011`, the depth is 4 and the parent node is `001`.

Value

A `HierarchicalPartition-class` object. Simply type object in the interactive R session to see which functions can be applied on it.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
## Not run:
set.seed(123)
m = cbind(rbind(matrix(rnorm(20*20, mean = 2, sd = 0.3), nr = 20),
                matrix(rnorm(20*20, mean = 0, sd = 0.3), nr = 20),
                matrix(rnorm(20*20, mean = 0, sd = 0.3), nr = 20)),
          rbind(matrix(rnorm(20*20, mean = 0, sd = 0.3), nr = 20),
                matrix(rnorm(20*20, mean = 1, sd = 0.3), nr = 20),
                matrix(rnorm(20*20, mean = 0, sd = 0.3), nr = 20)),
          rbind(matrix(rnorm(20*20, mean = 0, sd = 0.3), nr = 20),
                matrix(rnorm(20*20, mean = 0, sd = 0.3), nr = 20),
                matrix(rnorm(20*20, mean = 1, sd = 0.3), nr = 20))
         ) + matrix(rnorm(60*60, sd = 0.5), nr = 60)
rh = hierarchical_partition(m, top_value_method = "SD", partition_method = "kmeans")

## End(Not run)
```

`is_best_k-ConsensusPartition-method`

Test whether the current k is the best/optional k

Description

Test whether the current k is the best/optional k

Usage

```
## S4 method for signature 'ConsensusPartition'
is_best_k(object, k, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups.
`...`	Pass to `suggest_best_k,ConsensusPartition-method`.

Details

Optional best k is also assigned as `TRUE`.

Value

Logical scalar.

Examples

```
data(golub_cola)
obj = golub_cola["ATC", "skmeans"]
is_best_k(obj, k = 2)
is_best_k(obj, k = 3)
```

`is_best_k-ConsensusPartitionList-method`

Test whether the current k is the best/optional k

Description

Test whether the current k is the best/optional k

Usage

```
## S4 method for signature 'ConsensusPartitionList'
is_best_k(object, k, ...)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`k`	Number of subgroups.
`...`	Pass to `suggest_best_k,ConsensusPartitionList-method`.

Details

It tests on the partitions for every method.

Value

Logical vector.

Examples

```
data(golub_cola)
is_best_k(golub_cola, k = 3)
```

`is_best_k-dispatch` *Method dispatch page for is_best_k*

Description

Method dispatch page for `is_best_k`.

Dispatch

`is_best_k` can be dispatched on following classes:

- `is_best_k,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `is_best_k,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method

Examples

```
# no example
NULL
```

`is_leaf_node-HierarchicalPartition-method`

Test whether a node is a leaf node

Description

Test whether a node is a leaf node

Usage

```
## S4 method for signature 'HierarchicalPartition'
is_leaf_node(object, node, merge_node = merge_node_param())
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`node`	A vector of node IDs.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.

Examples

```
data(golub_cola_rh)
is_leaf_node(golub_cola_rh, all_leaves(golub_cola_rh))
```

`is_stable_k-ConsensusPartition-method`

Test whether the current k corresponds to a stable partition

Description

Test whether the current k corresponds to a stable partition

Usage

```
## S4 method for signature 'ConsensusPartition'
is_stable_k(object, k, stable_PAC = 0.1, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups.
`stable_PAC`	Cutoff for stable PAC.
`...`	Pass to `suggest_best_k,ConsensusPartition-method`.

Details

if 1-PAC for the k is larger than 0.9 (10% ambiguity for the partition), cola marks it as a stable partition.

Value

Logical scalar.

Examples

```
data(golub_cola)
obj = golub_cola["ATC", "skmeans"]
is_stable_k(obj, k = 2)
is_stable_k(obj, k = 3)
```

`is_stable_k-ConsensusPartitionList-method` *Test whether the current k corresponds to a stable partition*

Description

Test whether the current k corresponds to a stable partition

Usage

```
## S4 method for signature 'ConsensusPartitionList'
is_stable_k(object, k, ...)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`k`	Number of subgroups.
`...`	Pass to `suggest_best_k,ConsensusPartitionList-method`.

Details

It tests on the partitions for every method.

Value

Logical vector

Examples

```
data(golub_cola)
is_stable_k(golub_cola, k = 3)
```

`is_stable_k-dispatch` *Method dispatch page for is_stable_k*

Description

Method dispatch page for `is_stable_k`.

Dispatch

`is_stable_k` can be dispatched on following classes:

- `is_stable_k,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `is_stable_k,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

`knee_finder2` *Find the knee/elbow of a list of sorted points*

Description

Find the knee/elbow of a list of sorted points

Usage

```
knee_finder2(x, plot = FALSE)
```

Arguments

`x`	A numeric vector.
`plot`	Whether to make the plot.

Value

A vector of two numeric values. One for the left knee and the second for the right knee.

Examples

```
x = rnorm(1000)
knee_finder2(x, plot = TRUE)
```

`knitr_add_tab_item` *Add JavaScript tab in the report*

Description

Add JavaScript tab in the report

Usage

```
knitr_add_tab_item(code, header, prefix, desc = "", opt = NULL,
    message = NULL, hide_and_show = FALSE)
```

Arguments

`code`	R code to execute.
`header`	Header or the title for the tab.
`prefix`	Prefix of the chunk label.
`desc`	Decription in the tab.
`opt`	Options for the knitr chunk.
`message`	Message to print.
`hide_and_show`	Whether to hide the code output.

Details

Each tab contains the R source code and results generated from it (figure, tables, text, ...).

This function is only for internal use.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`knitr_insert_tabs` produces a complete HTML fragment.

Examples

```
# There is no example
NULL
```

`knitr_insert_tabs` *Generate the HTML fragment for the JavaScript tabs*

Description

Generate the HTML fragment for the JavaScript tabs

Usage

```
knitr_insert_tabs(uid)
```

Arguments

`uid`	A unique identifier for the div.

Details

The jQuery UI is used to generate html tabs (`https://jqueryui.com/tabs/`).

`knitr_insert_tabs` should be used after several calls of `knitr_add_tab_item` to generate a complete HTML fragment for all tabs with all necessary Javascript and css code.

This function is only for internal use.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`map_to_entrez_id` *Map to Entrez IDs*

Description

Map to Entrez IDs

Usage

```
map_to_entrez_id(from, org_db = "org.Hs.eg.db")
```

Arguments

`from`	The input gene ID type. Valid values should be in, e.g. `columns(org.Hs.eg.db::org.Hs.eg.db)`.
`org_db`	The annotation database.

Details

If there are multiple mappings from the input ID type to an unique Entrez ID, it randomly picks one.

Value

A named vectors where names are IDs with input ID type and values are the Entrez IDs.

The returned object normally is used in `functional_enrichment`.

Examples

```
map = map_to_entrez_id("ENSEMBL")
head(map)
```

`max_depth-HierarchicalPartition-method`

Max depth of the hierarchy

Description

Max depth of the hierarchy

Usage

```
## S4 method for signature 'HierarchicalPartition'
max_depth(object)
```

Arguments

`object`	A `HierarchicalPartition-class` object.

Value

A numeric value.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
max_depth(golub_cola_rh)
```

---

membership_heatmap-ConsensusPartition-method
Heatmap of membership in each partition

Description

Heatmap of membership in each partition

Usage

```
## S4 method for signature 'ConsensusPartition'
membership_heatmap(object, k, internal = FALSE,
    anno = object@anno, anno_col = get_anno_col(object),
    show_column_names = FALSE, column_names_gp = gpar(fontsize = 8), ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups.
`internal`	Used internally.
`anno`	A data frame of annotations for the original matrix columns. By default it uses the annotations specified in `consensus_partition` or `run_all_consensus_partition_methods`.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If `anno` is a data frame, `anno_col` should be a named list where names correspond to the column names in `anno`.
`show_column_names`	Whether show column names in the heatmap (which is the column name in the original matrix).
`column_names_gp`	Graphics parameters for column names.
`...`	Other arguments.

Details

Each row in the heatmap is the membership in one single partition.

Heatmap is split on rows by `top_n`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
membership_heatmap(golub_cola["ATC", "skmeans"], k = 3)
```

`merge_node-HierarchicalPartition-method` *Merge node*

Description

Merge node

Usage

```
## S4 method for signature 'HierarchicalPartition'
merge_node(object, node_id)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`node_id`	A vector of node IDs where each node is merged as a leaf node.

Value

A `HierarchicalPartition-class` object.

Examples

```
# There is no example
NULL
```

`merge_node_param` *Parameters to merge branches in subgroup dendrogram.*

Description

Parameters to merge branches in subgroup dendrogram.

Usage

```
merge_node_param(depth = Inf, min_n_signatures = -Inf,
    min_p_signatures = -Inf)
```

Arguments

`depth`	Depth of the dendrogram.
`min_n_signatures`	Minimal number of signatures for the partitioning on each node.
`min_p_signatures`	Minimal fraction of sigatures compared to the total number of rows on each node.

Examples

```
# There is no example
NULL
```

`ncol-ConsensusPartition-method`

Number of columns in the matrix

Description

Number of columns in the matrix

Usage

```
## S4 method for signature 'ConsensusPartition'
ncol(x)
```

Arguments

`x`	A `ConsensusPartition-class` object.

Examples

```
# There is no example
NULL
```

`ncol-ConsensusPartitionList-method` *Number of columns in the matrix*

Description

Number of columns in the matrix

Usage

```
## S4 method for signature 'ConsensusPartitionList'
ncol(x)
```

Arguments

`x`	A `ConsensusPartitionList-class` object.

Examples

```
# There is no example
NULL
```

`ncol-dispatch` *Method dispatch page for ncol*

Description

Method dispatch page for `ncol`.

Dispatch

`ncol` can be dispatched on following classes:

- `ncol,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `ncol,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `ncol,DownSamplingConsensusPartition-method`, `DownSamplingConsensusPartition-class` class method
- `ncol,HierarchicalPartition-method`, `HierarchicalPartition-class` class method

Examples

```
# no example
NULL
```

`ncol-DownSamplingConsensusPartition-method`

Number of columns in the matrix

Description

Number of columns in the matrix

Usage

```
## S4 method for signature 'DownSamplingConsensusPartition'
ncol(x)
```

Arguments

`x`	A `DownSamplingConsensusPartition-class` object.

Examples

```
# There is no example
NULL
```

`ncol-HierarchicalPartition-method`

Number of columns in the matrix

Description

Number of columns in the matrix

Usage

```
## S4 method for signature 'HierarchicalPartition'
ncol(x)
```

Arguments

`x`	A `HierarchicalPartition-class` object.

Examples

```
# There is no example
NULL
```

---

node_info-HierarchicalPartition-method

Information on the nodes

Description

Information on the nodes

Usage

```
## S4 method for signature 'HierarchicalPartition'
node_info(object)
```

Arguments

`object`	A `HierarchicalPartition-class` object.

Details

It returns the following node-level information:

id Node id.

n_columns Number of columns.

n_signatures Number of signatures.

p_signatures Percent of signatures.

is_leaf Whether the node is a leaf

Examples

```
# There is no example
NULL
```

---

node_level-HierarchicalPartition-method

Information on the nodes

Description

Information on the nodes

Usage

```
## S4 method for signature 'HierarchicalPartition'
node_level(object)
```

Arguments

`object`	A `HierarchicalPartition-class` object.

Details

It is the same as `node_info,HierarchicalPartition-method`.

Examples

```
# There is no example
NULL
```

---

nrow-ConsensusPartition-method

Number of rows in the matrix

Description

Number of rows in the matrix

Usage

```
## S4 method for signature 'ConsensusPartition'
nrow(x)
```

Arguments

`x`	A `ConsensusPartition-class` object.

Examples

```
# There is no example
NULL
```

`nrow-ConsensusPartitionList-method` *Number of rows in the matrix*

Description

Number of rows in the matrix

Usage

```
## S4 method for signature 'ConsensusPartitionList'
nrow(x)
```

Arguments

`x`	A `ConsensusPartitionList-class` object.

Examples

```
# There is no example
NULL
```

`nrow-dispatch` *Method dispatch page for nrow*

Description

Method dispatch page for `nrow`.

Dispatch

`nrow` can be dispatched on following classes:

- `nrow,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `nrow,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `nrow,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

`nrow-HierarchicalPartition-method` *Number of rows in the matrix*

Description

Number of rows in the matrix

Usage

```
## S4 method for signature 'HierarchicalPartition'
nrow(x)
```

Arguments

`x`	A `HierarchicalPartition-class` object.

Examples

```
# There is no example
NULL
```

`PAC` *The proportion of ambiguous clustering (PAC score)*

Description

The proportion of ambiguous clustering (PAC score)

Usage

```
PAC(consensus_mat, x1 = 0.1, x2 = 0.9, class = NULL)
```

Arguments

`consensus_mat`	A consensus matrix.
`x1`	Lower bound to define "ambiguous clustering".
`x2`	Upper bound to define "ambihuous clustering".
`class`	Subgroup labels. If it is provided, samples with silhouette score less than the 5^th percential are removed from PAC calculation.

Details

The PAC score is defined as F(x2) - F(x1) where F(x) is the CDF of the consensus matrix.

Value

A single numeric vaule.

See

See `https://www.nature.com/articles/srep06207` for explanation of PAC score.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
PAC(get_consensus(golub_cola[1, 1], k = 2))
PAC(get_consensus(golub_cola[1, 1], k = 3))
PAC(get_consensus(golub_cola[1, 1], k = 4))
PAC(get_consensus(golub_cola[1, 1], k = 5))
PAC(get_consensus(golub_cola[1, 1], k = 6))

# with specifying `class`
PAC(get_consensus(golub_cola[1, 1], k = 2),
    class = get_classes(golub_cola[1, 1], k = 2)[, 1])
```

`plot_ecdf-ConsensusPartition-method`
Plot the empirical cumulative distribution (eCDF) curve of the consensus matrix

---

Description

Plot the empirical cumulative distribution (eCDF) curve of the consensus matrix

Usage

```
## S4 method for signature 'ConsensusPartition'
plot_ecdf(object, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`...`	Other arguments.

Details

It plots eCDF curve for each k.

This function is mainly used in `collect_plots` and `select_partition_number` functions.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

See `ecdf` for a detailed explanation of the empirical cumulative distribution function.

Examples

```
data(golub_cola)
plot_ecdf(golub_cola["ATC", "skmeans"])
```

`predict_classes-ConsensusPartition-method`

Predict classes for new samples based on cola classification

Description

Predict classes for new samples based on cola classification

Usage

```
## S4 method for signature 'ConsensusPartition'
predict_classes(object, k, mat,
    silhouette_cutoff = 0.5,
    fdr_cutoff = cola_opt$fdr_cutoff,
    group_diff = cola_opt$group_diff,
    scale_rows = object@scale_rows,
    diff_method = "Ftest",
    method = "centroid",
    dist_method = c("euclidean", "correlation", "cosine"), nperm = 1000,
    p_cutoff = 0.05, plot = TRUE, col_fun = NULL,
    split_by_sigatures = FALSE, force = FALSE,
    verbose = TRUE, help = TRUE, prefix = "",
    mc.cores = 1, cores = mc.cores)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups to get the classifications.
`mat`	The new matrix where the sample classes are going to be predicted. The number of rows should be the same as the original matrix for cola analysis (also make sure the row orders are the same). Be careful that the scaling of `mat` should be the same as that applied in cola analysis.

`silhouette_cutoff`	Send to `get_signatures,ConsensusPartition-method` for determining signatures.
`fdr_cutoff`	Send to `get_signatures,ConsensusPartition-method` for determining signatures.
`group_diff`	Send to `get_signatures,ConsensusPartition-method` for determining signatures.
`scale_rows`	Send to `get_signatures,ConsensusPartition-method` for determining signatures.
`diff_method`	Send to `get_signatures,ConsensusPartition-method` for determining signatures.
`method`	Method for predicting class labels. Possible values are "centroid", "svm" and "randomForest".
`dist_method`	Distance method. Value should be "euclidean", "correlation" or "cosine". Send to `predict_classes,matrix-method`.
`nperm`	Number of permutatinos. It is used when `dist_method` is set to "euclidean" or "cosine". Send to `predict_classes,matrix-method`.
`p_cutoff`	Cutoff for the p-values for determining class assignment. Send to `predict_classes,matrix-method`.
`plot`	Whether to draw the plot that visualizes the process of prediction. Send to `predict_classes,matrix-method`.
`col_fun`	A color mapping function generated from `colorRamp2`. It is set to both heatmaps.
`split_by_sigatures`	Should the heatmaps be split based on k-means on the main heatmap, or on the patterns of the signature heatmap.
`force`	If the value is `TRUE` and when `get_signatures,ConsensusPartition-method` internally failed, top 1000 rows with the highest between-group mean difference are used for constructing the signature centroid matrix. It is basically used internally.
`verbose`	Whether to print messages. Send to `predict_classes,matrix-method`.
`help`	Whether to print help messages.
`prefix`	Used internally.
`mc.cores`	Number of cores. This argument will be removed in future versions.
`cores`	Number of cores, or a `cluster` object returned by `makeCluster`.

Details

The prediction is based on the signature centroid matrix from cola classification. The processes are as follows:

1. For the provided `ConsensusPartition-class` object and a selected k, the signatures that discriminate classes are extracted by `get_signatures,ConsensusPartition-method`. If number of signatures is more than 2000, only 2000 signatures are randomly sampled. 2. The signature centroid matrix is a k-column matrix where each column is the centroid of samples in the corresponding class, i.e. the mean across samples. If rows were scaled in cola analysis, the signature centroid matrix is the mean of scaled values and vise versa. Please note the samples with silhouette score less

than `silhouette_cutoff` are removed for calculating the centroids. 3. With the signature centroid matrix and the new matrix, it calls `predict_classes,matrix-method` to perform the prediction. Please see more details of the prediction on that help page. Please note, the scales of the new matrix should be the same as the matrix used for cola analysis.

Value

A data frame with two columns: the class labels (in numeric) and the corresponding p-values.

See Also

`predict_classes,matrix-method` that predicts the classes for new samples.

Examples

```
data(golub_cola)
res = golub_cola["ATC:skmeans"]
mat = get_matrix(res)
# note scaling should be applied here because the matrix was scaled in the cola analysis
mat2 = t(scale(t(mat)))
cl = predict_classes(res, k = 3, mat2)
# compare the real classification and the predicted classification
data.frame(cola_class = get_classes(res, k = 3)[, "class"],
           predicted = cl[, "class"])
# change to correlation method
cl = predict_classes(res, k = 3, mat2, dist_method = "correlation")
# compare the real classification and the predicted classification
data.frame(cola_class = get_classes(res, k = 3)[, "class"],
           predicted = cl[, "class"])
```

`predict_classes-dispatch`

Method dispatch page for predict_classes

Description

Method dispatch page for `predict_classes`.

Dispatch

`predict_classes` can be dispatched on following classes:

- `predict_classes,matrix-method`, `matrix-class` class method
- `predict_classes,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

`predict_classes-matrix-method`

Predict classes for new samples based on signature centroid matrix

Description

Predict classes for new samples based on signature centroid matrix

Usage

```
## S4 method for signature 'matrix'
predict_classes(object, mat, dist_method = c("euclidean", "correlation", "cosine"),
   nperm = 1000, p_cutoff = 0.05, plot = TRUE, col_fun = NULL, split_by_sigatures = FALSE,
   verbose = TRUE, prefix = "", mc.cores = 1, cores = mc.cores, width1 = NULL, width2 = NULL)
```

Arguments

`object`	The signature centroid matrix. See the Details section.
`mat`	The new matrix where the classes are going to be predicted. The number of rows should be the same as the signature centroid matrix (also make sure the row orders are the same). Be careful that `mat` should be in the same scale as the centroid matrix.
`dist_method`	Distance method. Value should be "euclidean", "correlation" or "cosine".
`nperm`	Number of permutatinos. It is used when `dist_method` is set to "euclidean" or "cosine".
`p_cutoff`	Cutoff for the p-values for determining class assignment.
`plot`	Whether to draw the plot that visualizes the process of prediction.
`col_fun`	A color mapping function generated from `colorRamp2`. It is set to both heatmaps.
`verbose`	Whether to print messages.
`split_by_sigatures`	Should the heatmaps be split based on k-means on the main heatmap, or on the patterns of the signature heatmap.
`prefix`	Used internally.
`mc.cores`	Number of cores. This argument will be removed in future versions.
`cores`	Number of cores, or a `cluster` object returned by `makeCluster`.
`width1`	Width of the first heatmap.
`width2`	Width of the second heatmap.

Details

The signature centroid matrix is a k-column matrix where each column is the centroid of samples in the corresponding class (k-group classification).

For each sample in the new matrix, the task is basically to test which signature centroid the current sample is the closest to. There are two methods: the Euclidean distance and the correlation (Spearman) distance.

For the Euclidean/cosine distance method, for the vector denoted as x which corresponds to sample i in the new matrix, to test which class should be assigned to sample i, the distance between sample i and all k signature centroids are calculated and denoted as d_1, d_2, ..., d_k. The class with the smallest distance is assigned to sample i. The distances for k centroids are sorted increasingly, and we design a statistic named "difference ratio", denoted as r and calculated as: (|d_(1) - d_(2)|)/mean(d), which is the difference between the smallest distance and the second smallest distance, normalized by the mean distance. To test the statistical significance of r, we randomly permute rows of the signature centroid matrix and calculate r_rand. The random permutation is performed `n_perm` times and the p-value is calculated as the proportion of r_rand being larger than r.

For the correlation method, the distance is calculated as the Spearman correlation between sample i and signature centroid k. The label for the class with the maximal correlation value is assigned to sample i. The p-value is simply calculated by `cor.test` between sample i and centroid k.

If a sample is tested with a p-value higher than `p_cutoff`, the corresponding class label is set to `NA`.

Value

A data frame with two columns: the class labels (the column names of the signature centroid matrix are treated as class labels) and the corresponding p-values.

Examples

```
data(golub_cola)
res = golub_cola["ATC:skmeans"]
mat = get_matrix(res)
# note scaling should be applied here because the matrix was scaled in the cola analysis
mat2 = t(scale(t(mat)))

tb = get_signatures(res, k = 3, plot = FALSE)
sig_mat = tb[, grepl("scaled_mean", colnames(tb))]
sig_mat = as.matrix(sig_mat)
colnames(sig_mat) = paste0("class", seq_len(ncol(sig_mat)))
# this is how the signature centroid matrix looks like:
head(sig_mat)

mat2 = mat2[tb$which_row, , drop = FALSE]

# now we predict the class for `mat2` based on `sig_mat`
predict_classes(sig_mat, mat2)
```

`print.hc_table_suggest_best_k` *Print the hc_table_suggest_best_k object*

Description

Print the hc_table_suggest_best_k object

Usage

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

Arguments

`x`	A `hc_table_suggest_best_k` object from `suggest_best_k,HierarchicalPartition-method`.
`...`	Other arguments.

Examples

```
# There is no example
NULL
```

`recalc_stats` *Recalculate statistics in the ConsensusPartitionList object*

Description

Recalculate statistics in the ConsensusPartitionList object

Usage

```
recalc_stats(rl)
```

Arguments

`rl`	A `ConsensusPartitionList-class` object.

Details

It updates the `stat` slot in the ConsensusPartitionList object, used internally.

Examples

```
# There is no example
NULL
```

`register_NMF` *Register NMF partitioning method*

Description

Register NMF partitioning method

Usage

```
register_NMF()
```

Details

NMF analysis is performed by `nmf`.

Examples

```
# There is no example
NULL
```

`register_partition_methods` *Register user-defined partitioning methods*

Description

Register user-defined partitioning methods

Usage

```
register_partition_methods(..., scale_method = c("z-score", "min-max", "none"))
```

Arguments

`...`	A named list of functions.
`scale_method`	Normally, data matrix is scaled by rows before sent to the partition function. The default scaling is applied by `scale`. However, some partition functions may not accept negative values which are produced by `scale`. Here `scale_method` can be set to `min-max` which scales rows by `(x - min)/(max - min)`. Note here `scale_method` only means the method to scale rows. When `scale_rows` is set to `FALSE` in `consensus_partition` or `run_all_consensus_partition_methods`, there will be no row scaling when doing partitioning. The value for `scale_method` can be a vector if user specifies more than one partition function.

Details

The user-defined function should accept at least two arguments. The first two arguments are the data matrix and the number of subgroups. The third optional argument should always be `...` so that parameters for the partition function can be passed by `partition_param` from `consensus_partition`. If users forget to add `...`, it is added internally.

The function should return a vector of partitions (or class labels) or an object which can be recognized by `cl_membership`.

The partition function should be applied on columns (Users should be careful with this because some R functions apply on rows and some R functions apply on columns). E.g. following is how we register `kmeans` partition method:

```
register_partition_methods(
    kmeans = function(mat, k, ...) {
        # mat is transposed because kmeans() applies on rows
        kmeans(t(mat), centers = k, ...)$centers
    }
)
```

The registered partitioning methods will be used as defaults in `run_all_consensus_partition_methods`.

To remove a partitioning method, use `remove_partition_methods`.

There are following default partitioning methods:

"hclust" hierarchcial clustering with Euclidean distance, later columns are partitioned by `cutree`. If users want to use another distance metric or clustering method, consider to register a new partitioning method. E.g. `register_partition_methods(hclust_cor = function(mat, k) cutree(hclust(as.dist(cor(mat)))))`.

"kmeans" by `kmeans`.

"skmeans" by `skmeans`.

"pam" by `pam`.

"mclust" by `Mclust`. mclust is applied to the first three principle components from rows.

Users can register two other pre-defined partitioning methods by `register_NMF` and `register_SOM`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`all_partition_methods` lists all registered partitioning methods.

Examples

```
all_partition_methods()
register_partition_methods(
    random = function(mat, k) sample(k, ncol(mat), replace = TRUE)
)
all_partition_methods()
remove_partition_methods("random")
```

`register_SOM` *Register SOM partitioning method*

Description

Register SOM partitioning method

Usage

```
register_SOM()
```

Details

The SOM analysis is performed by `som`.

Examples

```
# There is no example
NULL
```

`register_top_value_methods` *Register user-defined top-value methods*

Description

Register user-defined top-value methods

Usage

```
register_top_value_methods(..., validate = TRUE)
```

Arguments

`...`	A named list of functions.
`validate`	Whether validate the functions.

Details

The user-defined function should accept one argument which is the data matrix where the scores are calculated by rows. Rows with top scores are treated as "top rows" in cola analysis. Following is how we register "SD" (standard deviation) top-value method:

```
register_top_value_methods(SD = function(mat) apply(mat, 1, sd))
```

Of course, you can use `rowSds` to give a faster calculation of row SD:

```
register_top_value_methods(SD = rowSds)
```

The registered top-value method will be used as defaults in `run_all_consensus_partition_methods`.

To remove a top-value method, use `remove_top_value_methods`.

There are four default top-value methods:

- **"SD"** standard deviation, by `rowSds`.
- **"CV"** coefficient variance, calculated as `sd/(mean+s)` where `s` is the 10^th percentile of all row means.
- **"MAD"** median absolute deviation, by `rowMads`.
- **"ATC"** the `ATC` method.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`all_top_value_methods` lists all registered top-value methods.

Examples

```
all_top_value_methods()
register_top_value_methods(
    ATC_spearman = function(mat) ATC(mat, method = "spearman")
)
all_top_value_methods()
remove_top_value_methods("ATC_spearman")
```

`relabel_class` *Relabel class labels according to the reference labels*

Description

Relabel class labels according to the reference labels

Usage

```
relabel_class(class, ref, full_set = union(class, ref), return_map = TRUE)
```

Arguments

`class`	A vector of class labels.
`ref`	A vector of reference labels.
`full_set`	The full set of labels.
`return_map`	Whether to return the mapping of the adjusted labels.

Details

In partitions, the exact value of the class label is not of importance. E.g. for two partitions `a, a, a, b, b, b, b` and `b, b, b, a, a, a, a`, they are the same partitions although the labels of `a` and `b` are switched in the two partitions. Even the partition `c, c, c, d, d, d, d` is the same as the previous two although it uses a different set of labels. Here `relabel_class` function relabels `class` vector according to the labels in `ref` vector by looking for a mapping `m()` to maximize `sum(m(class) == ref)`.

Mathematically, this is called linear sum assignment problem and it is solved by `solve_LSAP`.

Value

A named vector where names correspond to the labels in `class` and values correspond to `ref`, which means `map = relabel_class(class, ref); map[class]` returns the relabelled labels.

The returned object attaches a data frame with three columns:

- original labels. in `class`
- adjusted labels. according to `ref`
- reference labels. in `ref`

If `return_map` in the `relabel_class` is set to `FALSE`, the function simply returns a vector of adjusted class labels.

If the function returns the mapping vector (when `return_map = TRUE`), the mapping variable is always character, which means, if your `class` and `ref` are numeric, you need to convert them back to numeric explicitely. If `return_map = FALSE`, the returned relabelled vector has the same mode as `class`.

Examples

```
class = c(rep("a", 10), rep("b", 3))
ref = c(rep("b", 4), rep("a", 9))
relabel_class(class, ref)
relabel_class(class, ref, return_map = FALSE)
# if class and ref are from completely different sets
class = c(rep("A", 10), rep("B", 3))
relabel_class(class, ref)

# class labels are numeric
class = c(rep(1, 10), rep(2, 3))
ref = c(rep(2, 4), rep(1, 9))
relabel_class(class, ref)
relabel_class(class, ref, return_map = FALSE)
```

`remove_partition_methods`

Remove partitioning methods

Description

Remove partitioning methods

Usage

```
remove_partition_methods(method)
```

Arguments

`method`	Name of the partitioning methods to be removed.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`remove_top_value_methods`

Remove top-value methods

Description

Remove top-value methods

Usage

```
remove_top_value_methods(method)
```

Arguments

`method`	Name of the top-value methods to be removed.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`rownames-ConsensusPartition-method`

Row names of the matrix

Description

Row names of the matrix

Usage

```
## S4 method for signature 'ConsensusPartition'
rownames(x)
```

Arguments

`x`	A `ConsensusPartition-class` object.

Examples

```
# There is no example
NULL
```

`rownames-ConsensusPartitionList-method` *Row names of the matrix*

Description

Row names of the matrix

Usage

```
## S4 method for signature 'ConsensusPartitionList'
rownames(x)
```

Arguments

`x`	A `ConsensusPartitionList-class` object.

Examples

```
# There is no example
NULL
```

`rownames-dispatch` *Method dispatch page for rownames*

Description

Method dispatch page for `rownames`.

Dispatch

`rownames` can be dispatched on following classes:

- `rownames,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `rownames,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `rownames,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method

Examples

```
# no example
NULL
```

`rownames-HierarchicalPartition-method`

Row names of the matrix

Description

Row names of the matrix

Usage

```
## S4 method for signature 'HierarchicalPartition'
rownames(x)
```

Arguments

`x`	A `HierarchicalPartition-class` object.

Examples

```
# There is no example
NULL
```

`run_all_consensus_partition_methods`

Consensus partitioning for all combinations of methods

Description

Consensus partitioning for all combinations of methods

Usage

```
run_all_consensus_partition_methods(data,
    top_value_method = all_top_value_methods(),
    partition_method = all_partition_methods(),
    max_k = 6, k = NULL,
    top_n = NULL,
    mc.cores = 1, cores = mc.cores, anno = NULL, anno_col = NULL,
    sample_by = "row", p_sampling = 0.8, partition_repeat = 50,
    scale_rows = NULL, verbose = TRUE, help = cola_opt$help)
```

Arguments

`data`	A numeric matrix where subgroups are found by columns.
`top_value_method`	Method which are used to extract top n rows. Allowed methods are in `all_top_value_methods` and can be self-added by `register_top_value_methods`.
`partition_method`	Method which are used to partition samples. Allowed methods are in `all_partition_methods` and can be self-added by `register_partition_methods`.
`max_k`	Maximal number of subgroups to try. The function will try `2:max_k` subgroups.
`k`	Alternatively, you can specify a vector k.
`top_n`	Number of rows with top values. The value can be a vector with length > 1. When n > 5000, the function only randomly sample 5000 rows from top n rows. If `top_n` is a vector, paritition will be applied to every values in `top_n` and consensus partition is summarized from all partitions.
`mc.cores`	Number of cores to use. This argument will be removed in future versions.
`cores`	Number of cores, or a `cluster` object returned by `makeCluster`.
`anno`	A data frame with known annotation of columns.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If `anno` is a data frame, `anno_col` should be a named list where names correspond to the column names in `anno`.
`sample_by`	Should randomly sample the matrix by rows or by columns?
`p_sampling`	Proportion of the top n rows to sample.
`partition_repeat`	Number of repeats for the random sampling.
`scale_rows`	Whether to scale rows. If it is `TRUE`, scaling method defined in `register_partition_methods` is used.
`verbose`	Whether to print messages.
`help`	Whether to print help messages.

Details

The function performs consensus partitioning by `consensus_partition` for all combinations of top-value methods and partitioning methods.

It also adjsuts the subgroup labels for all methods and for all k to make them as consistent as possible.

Value

A `ConsensusPartitionList-class` object. Simply type object in the interactive R session to see which functions can be applied on it.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
## Not run:
set.seed(123)
m = cbind(rbind(matrix(rnorm(20*20, mean = 1), nr = 20),
                matrix(rnorm(20*20, mean = -1), nr = 20)),
          rbind(matrix(rnorm(20*20, mean = -1), nr = 20),
                matrix(rnorm(20*20, mean = 1), nr = 20))
         ) + matrix(rnorm(40*40), nr = 40)
rl = run_all_consensus_partition_methods(data = m, top_n = c(20, 30, 40))

## End(Not run)
```

`select_partition_number-ConsensusPartition-method`

Several plots for determining the optimized number of subgroups

Description

Several plots for determining the optimized number of subgroups

Usage

```
## S4 method for signature 'ConsensusPartition'
select_partition_number(object, mark_best = TRUE, all_stats = FALSE)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`mark_best`	Whether mark the best k in the plot.
`all_stats`	Whether to show all statistics that were calculated. Used internally.

Details

There are following plots made:

- eCDF of the consensus matrix under each k, made by `plot_ecdf,ConsensusPartition-method`,
- `PAC` score,
- mean sihouette score,
- the `concordance` for each partition to the consensus partition,
- area increase of the area under the ECDF of consensus matrix with increasing k,
- Rand index for current k compared to k - 1,
- Jaccard coefficient for current k compared to k - 1,

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
select_partition_number(golub_cola["ATC", "skmeans"])
```

`show-ConsensusPartition-method`

Print the ConsensusPartition object

Description

Print the ConsensusPartition object

Usage

```
## S4 method for signature 'ConsensusPartition'
show(object)
```

Arguments

`object`	A `ConsensusPartition-class` object.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`show-ConsensusPartitionList-method` *Print the ConsensusPartitionList object*

Description

Print the ConsensusPartitionList object

Usage

```
## S4 method for signature 'ConsensusPartitionList'
show(object)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

`show-dispatch` *Method dispatch page for show*

Description

Method dispatch page for `show`.

Dispatch

`show` can be dispatched on following classes:

- `show,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `show,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `show,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `show,DownSamplingConsensusPartition-method`, `DownSamplingConsensusPartition-class` class method

Examples

```
# no example
NULL
```

---

show-DownSamplingConsensusPartition-method

Print the DownSamplingConsensusPartition object

Description

Print the DownSamplingConsensusPartition object

Usage

```
## S4 method for signature 'DownSamplingConsensusPartition'
show(object)
```

Arguments

`object`	A `DownSamplingConsensusPartition-class` object.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_ds)
golub_cola_ds
```

`show-HierarchicalPartition-method`

Print the HierarchicalPartition object

Description

Print the HierarchicalPartition object

Usage

```
## S4 method for signature 'HierarchicalPartition'
show(object)
```

Arguments

`object`	a `HierarchicalPartition-class` object

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
golub_cola_rh
```

`split_node-HierarchicalPartition-method`

Split node

Description

Split node

Usage

```
## S4 method for signature 'HierarchicalPartition'
split_node(object, node_id,
    subset = object@param$subset,
   min_samples = object@param$min_samples, max_k = object@param$max_k, cores = object@param$cores,
    verbose = TRUE,
   top_n = object@param$top_n, min_n_signatures = object@param$min_n_signatures,
    group_diff = object@param$group_diff, fdr_cutoff = object@param$fdr_cutoff)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`node_id`	A single ID of a node that is going to be split.
`subset`	The same as in `hierarchical_partition`.
`min_samples`	The same as in `hierarchical_partition`.
`max_k`	max_k The same as in `hierarchical_partition`.
`cores`	Number of cores.
`verbose`	Whether to print messages.
`top_n`	The same as in `hierarchical_partition`.
`min_n_signatures`	The same as in `hierarchical_partition`.
`group_diff`	The same as in `hierarchical_partition`.
`fdr_cutoff`	The same as in `hierarchical_partition`.

Details

It applies hierarchical consensus partitioning on the specified node.

Value

A `HierarchicalPartition-class` object.

Examples

```
# There is no example
NULL
```

---

suggest_best_k-ConsensusPartition-method

Suggest the best number of subgroups

Description

Suggest the best number of subgroups

Usage

```
## S4 method for signature 'ConsensusPartition'
suggest_best_k(object,
    jaccard_index_cutoff = select_jaccard_cutoff(ncol(object)),
    mean_silhouette_cutoff = NULL,
    stable_PAC = 0.1, help = cola_opt$help)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`jaccard_index_cutoff`	The cutoff for Jaccard index for comparing to previous k.
`mean_silhouette_cutoff`	Cutoff for mean silhourtte scores.
`stable_PAC`	Cutoff for stable PAC. This argument only take effect when `mean_silhouette_cutoff` is set to `NULL`.
`help`	Whether to print help message.

Details

The best k is selected according to following rules:

- All k with Jaccard index larger than `jaccard_index_cutoff` are removed because increasing k does not provide enough extra information. If all k are removed, it is marked as no subgroup is detected.
- If all k with Jaccard index larger than 0.75, k with the highest mean silhourtte score is taken as the best k.
- For all k with mean silhouette score larger than `mean_silhouette_cutoff`, the maximal k is taken as the best k, and other k are marked as optional best k.
- If argument `mean_silhouette_cutoff` is set to NULL, which means we do not filter by mean silhouette scores while by 1-PAC scores. Similarly, k with the highest 1-PAC is taken the best k and other k are marked as optional best k.
- If it does not fit the second rule. The k with the maximal vote of the highest 1-PAC score, highest mean silhouette, and highest concordance is taken as the best k.

It should be noted that it is difficult to find the best k deterministically, we encourage users to compare results for all k and determine a proper one which best explain their studies.

Value

The best k.

See

The selection of the best k can be visualized by `select_partition_number`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
obj = golub_cola["ATC", "skmeans"]
suggest_best_k(obj)
```

`suggest_best_k-ConsensusPartitionList-method`

Suggest the best number of subgroups

Description

Suggest the best number of subgroups

Usage

```
## S4 method for signature 'ConsensusPartitionList'
suggest_best_k(object, jaccard_index_cutoff = select_jaccard_cutoff(ncol(object)))
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`jaccard_index_cutoff`	The cutoff for Jaccard index for comparing to previous k.

Details

It basically gives the best k for each combination of top-value method and partitioning method by calling `suggest_best_k,ConsensusPartition-method`.

1-PAC score higher than 0.95 is treated as very stable partition (marked by `**`) and higher than 0.9 is treated as stable partition (marked by `*`).

Value

A data frame with the best k and other statistics for each combination of methods.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
suggest_best_k(golub_cola)
```

---

suggest_best_k-dispatch

Method dispatch page for suggest_best_k

Description

Method dispatch page for `suggest_best_k`.

Dispatch

`suggest_best_k` can be dispatched on following classes:

- `suggest_best_k,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `suggest_best_k,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `suggest_best_k,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

---

suggest_best_k-HierarchicalPartition-method

Guess the best number of partitions

Description

Guess the best number of partitions

Usage

```
## S4 method for signature 'HierarchicalPartition'
suggest_best_k(object)
```

Arguments

`object`	A `HierarchicalPartition-class` object.

Details

It basically gives the best k at each node.

Value

A data frame with the best k and other statistics for each node.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
suggest_best_k(golub_cola_rh)
```

`test_between_factors` *Test whether a list of factors are correlated*

Description

Test whether a list of factors are correlated

Usage

```
test_between_factors(x, y = NULL, all_factors = FALSE, verbose = FALSE)
```

Arguments

`x`	A data frame or a vector which contains discrete or continuous variables. if `y` is omit, pairwise testing for all columns in `x` is performed.
`y`	A data frame or a vector which contains discrete or continuous variables.
`all_factors`	Are all columns in `x` and `y` enforced to be factors?
`verbose`	Whether to print messages.

Details

Pairwise test is applied to every two columns in the data frames. Methods are:

- two numeric variables: correlation test by `cor.test` is applied (Spearman method);
- two character or factor variables: `chisq.test` is applied;
- one numeric variable and one character/factor variable: oneway ANOVA test by `oneway.test` is applied.

This function can be used to test the correlation between the predicted classes and other known factors.

Value

A matrix of p-values. If there are NA values, basically it means there are no efficient data points to perform the test.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
df = data.frame(
    v1 = rnorm(100),
    v2 = sample(letters[1:3], 100, replace = TRUE),
    v3 = sample(LETTERS[5:6], 100, replace = TRUE)
)
test_between_factors(df)
x = runif(100)
test_between_factors(x, df)
```

---

test_to_known_factors-ConsensusPartition-method

Test correspondance between predicted subgroups and known factors

Description

Test correspondance between predicted subgroups and known factors

Usage

```
## S4 method for signature 'ConsensusPartition'
test_to_known_factors(object, k, known = get_anno(object),
    silhouette_cutoff = 0.5, verbose = FALSE)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of subgroups. It uses all k if it is not specified.
`known`	A vector or a data frame with known factors. By default it is the annotation table set in `consensus_partition` or `run_all_consensus_partition_methods`.
`silhouette_cutoff`	Cutoff for sihouette scores. Samples with value less than it are omit.
`verbose`	Whether to print messages.

Details

The test is performed by `test_between_factors` between the predicted classes and user's annotation table.

Value

A data frame with the following columns:

- number of samples used to test after filtered by `silhouette_cutoff`,
- p-values from the tests,
- number of subgroups.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
res = golub_cola["ATC:skmeans"]
anno = get_anno(res)
anno
test_to_known_factors(res, k = 3)
# or explicitly specify known argument
test_to_known_factors(res, k = 3, known = anno)
```

`test_to_known_factors-ConsensusPartitionList-method`

Test correspondance between predicted classes and known factors

Description

Test correspondance between predicted classes and known factors

Usage

```
## S4 method for signature 'ConsensusPartitionList'
test_to_known_factors(object, k, known = get_anno(object),
    silhouette_cutoff = 0.5, verbose = FALSE)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`k`	Number of subgroups. It uses all `k` if it is not set.
`known`	A vector or a data frame with known factors. By default it is the annotation table set in `consensus_partition` or `run_all_consensus_partition_methods`.
`silhouette_cutoff`	Cutoff for sihouette scores. Samples with value less than this are omit.
`verbose`	Whether to print messages.

Details

The function basically sends each `ConsensusPartition-class` object to `test_to_known_factors,ConsensusPartition-` and merges results afterwards.

Value

A data frame with the following columns:

- number of samples used to test after filtered by `silhouette_cutoff`,
- p-values from the tests,
- number of subgroups.

If there are NA values, basically it means there are no efficient data points to perform the test.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`test_between_factors`, `test_to_known_factors,ConsensusPartition-method`

Examples

```
data(golub_cola)
test_to_known_factors(golub_cola)
```

`test_to_known_factors-dispatch`

Method dispatch page for test_to_known_factors

Description

Method dispatch page for `test_to_known_factors`.

Dispatch

`test_to_known_factors` can be dispatched on following classes:

- `test_to_known_factors,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `test_to_known_factors,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `test_to_known_factors,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `test_to_known_factors,DownSamplingConsensusPartition-method`, `DownSamplingConsensusPartition-clas` class method

Examples

```
# no example
NULL
```

---

test_to_known_factors-DownSamplingConsensusPartition-method

Test correspondance between predicted subgroups and known factors

Description

Test correspondance between predicted subgroups and known factors

Usage

```
## S4 method for signature 'DownSamplingConsensusPartition'
test_to_known_factors(object, k, known = get_anno(object),
    p_cutoff = 0.05, verbose = FALSE)
```

Arguments

`object`	A `DownSamplingConsensusPartition-class` object.
`k`	Number of subgroups. It uses all k if it is not specified.
`known`	A vector or a data frame with known factors. By default it is the annotation table set in `consensus_partition_by_down_sampling`.
`p_cutoff`	Cutoff for p-values for the class prediction. Samples with p-value higher than it are omit.
`verbose`	Whether to print messages.

Details

The test is performed by `test_between_factors` between the predicted classes and user's annotation table.

Value

A data frame with the following columns:

- number of samples used to test after filtered by `p_cutoff`,
- p-values from the tests,
- number of subgroups.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_ds)
test_to_known_factors(golub_cola_ds, k = 3)
test_to_known_factors(golub_cola_ds)
```

`test_to_known_factors-HierarchicalPartition-method`

Test correspondance between predicted classes and known factors

Description

Test correspondance between predicted classes and known factors

Usage

```
## S4 method for signature 'HierarchicalPartition'
test_to_known_factors(object, known = get_anno(object[1]),
    merge_node = merge_node_param(), verbose = FALSE)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`merge_node`	Parameters to merge sub-dendrograms, see `merge_node_param`.
`known`	A vector or a data frame with known factors. By default it is the annotation table set in `hierarchical_partition`.
`verbose`	Whether to print messages.

Value

A data frame with columns:

- number of samples
- p-values from the tests
- number of classes

The classifications are extracted for each depth.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola_rh)
# golub_cola_rh already has known annotations, so test_to_known_factors()
# can be directly applied
test_to_known_factors(golub_cola_rh)
```

`top_elements_overlap` *Overlap of top elements from different metrics*

Description

Overlap of top elements from different metrics

Usage

```
top_elements_overlap(object, top_n = round(0.25*length(object[[1]])),
    method = c("euler", "upset", "venn", "correspondance"),
    fill = NULL, ...)
```

Arguments

`object`	A list which contains values from different metrics.
`top_n`	Number of top rows.
`method`	`euler`: plot Euler diagram by `euler`; `upset`: draw the Upset plot by `UpSet`; `venn`: plot Venn diagram by `venn`; `correspondance`: use `correspond_between_rankings`.
`fill`	Filled color for the Euler diagram. The value should be a color vector. Transparency of 0.5 are added internally.
`...`	Additional arguments passed to `plot.euler`, `UpSet` or `correspond_between_rankings`.

Details

The i^th value in every vectors in `object` should correspond to the same element from the original data.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
require(matrixStats)
set.seed(123)
mat = matrix(rnorm(1000), nrow = 100)
lt = list(sd = rowSds(mat), mad = rowMads(mat))
top_elements_overlap(lt, top_n = 20, method = "euler")
top_elements_overlap(lt, top_n = 20, method = "upset")
top_elements_overlap(lt, top_n = 20, method = "venn")
top_elements_overlap(lt, top_n = 20, method = "correspondance")
```

---

top_rows_heatmap-ConsensusPartition-method

Heatmap of top rows

Description

Heatmap of top rows

Usage

```
## S4 method for signature 'ConsensusPartition'
top_rows_heatmap(object, top_n = min(object@top_n), k = NULL,
    anno = get_anno(object), anno_col = get_anno_col(object),
    scale_rows = object@scale_rows, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`top_n`	Number of top rows.
`k`	Number of subgroups. If it is not specified, it uses the "best k".
`anno`	A data frame of annotations.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If `anno` is a data frame, `anno_col` should be a named list where names correspond to the column names in `anno`.
`scale_rows`	Wether to scale rows.
`...`	Pass to `top_rows_heatmap,matrix-method`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`top_rows_heatmap,matrix-method`

Examples

```
data(golub_cola)
top_rows_heatmap(golub_cola["ATC:skmeans"])
```

`top_rows_heatmap-ConsensusPartitionList-method`

Heatmap of top rows from different top-value methods

Description

Heatmap of top rows from different top-value methods

Usage

```
## S4 method for signature 'ConsensusPartitionList'
top_rows_heatmap(object, top_n = min(object@list[[1]]@top_n),
    anno = get_anno(object), anno_col = get_anno_col(object),
    scale_rows = object@list[[1]]@scale_rows, ...)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`top_n`	Number of top rows.
`anno`	A data frame of annotations for the original matrix columns. By default it uses the annotations specified in `run_all_consensus_partition_methods`.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If anno is a data frame, `anno_col` should be a named list where names correspond to the column names in anno.
`scale_rows`	Wether to scale rows.
`...`	Pass to `top_rows_heatmap,matrix-method`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`top_rows_heatmap,matrix-method`

Examples

```
data(golub_cola)
top_rows_heatmap(golub_cola)
```

---

top_rows_heatmap-dispatch

Method dispatch page for top_rows_heatmap

Description

Method dispatch page for `top_rows_heatmap`.

Dispatch

`top_rows_heatmap` can be dispatched on following classes:

- `top_rows_heatmap,ConsensusPartition-method`, `ConsensusPartition-class` class method
- `top_rows_heatmap,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `top_rows_heatmap,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `top_rows_heatmap,matrix-method`, `matrix-class` class method

Examples

```
# no example
NULL
```

---

top_rows_heatmap-HierarchicalPartition-method

Heatmap of top rows from different top-value methods

Description

Heatmap of top rows from different top-value methods

Usage

```
## S4 method for signature 'HierarchicalPartition'
top_rows_heatmap(object, top_n = min(object@list[[1]]@top_n),
    anno = get_anno(object), anno_col = get_anno_col(object),
    scale_rows = object@list[[1]]@scale_rows, ...)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`top_n`	Number of top rows.
`anno`	A data frame of annotations for the original matrix columns. By default it uses the annotations specified in `hierarchical_partition`.
`anno_col`	A list of colors (color is defined as a named vector) for the annotations. If `anno` is a data frame, `anno_col` should be a named list where names correspond to the column names in `anno`.
`scale_rows`	Wether to scale rows.
`...`	Pass to `top_rows_heatmap,matrix-method`

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`top_rows_heatmap,matrix-method`

Examples

```
# There is no example
NULL
```

---

top_rows_heatmap-matrix-method

Heatmap of top rows from different top-value methods

Description

Heatmap of top rows from different top-value methods

Usage

```
## S4 method for signature 'matrix'
top_rows_heatmap(object, all_top_value_list = NULL,
    top_value_method = all_top_value_methods(),
    bottom_annotation = NULL,
    top_n = round(0.25*nrow(object)), scale_rows = TRUE, ...)
```

Arguments

`object`	A numeric matrix.
`all_top_value_list`	Top-values that have already been calculated from the matrix. If it is `NULL` the values are calculated by methods in `top_value_method` argument.
`top_value_method`	Methods defined in `all_top_value_methods`.
`bottom_annotation`	A `HeatmapAnnotation-class` object.
`top_n`	Number of top rows to show in the heatmap.
`scale_rows`	Whether to scale rows.
`...`	Pass to `Heatmap`.

Details

The function makes heatmaps where the rows are scaled (or not scaled) for the top n rows from different top-value methods.

The top n rows are used for subgroup classification in cola analysis, so the heatmaps show which top-value method gives better candidate rows for the classification.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
set.seed(123)
mat = matrix(rnorm(1000), nrow = 100)
top_rows_heatmap(mat, top_n = 25)
```

---

top_rows_overlap-ConsensusPartitionList-method

Overlap of top rows from different top-value methods

Description

Overlap of top rows from different top-value methods

Usage

```
## S4 method for signature 'ConsensusPartitionList'
top_rows_overlap(object, top_n = min(object@list[[1]]@top_n),
    method = c("euler", "upset", "venn", "correspondance"), fill = NULL, ...)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`top_n`	Number of top rows.
`method`	`euler`: plot Euler diagram by `euler`; `upset`: draw the Upset plot by `UpSet`; `venn`: plot Venn diagram by `venn`; `correspondance`: use `correspond_between_rankings`.
`fill`	Filled color for the Euler diagram. The value should be a color vector. Transparency of 0.5 are added internally.
`...`	Additional arguments passed to `plot.euler`, `UpSet` or `correspond_between_rankings`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`top_elements_overlap`

Examples

```
data(golub_cola)
top_rows_overlap(golub_cola, method = "euler")
top_rows_overlap(golub_cola, method = "upset")
top_rows_overlap(golub_cola, method = "venn")
top_rows_overlap(golub_cola, method = "correspondance")
```

`top_rows_overlap-dispatch`

Method dispatch page for top_rows_overlap

Description

Method dispatch page for `top_rows_overlap`.

Dispatch

`top_rows_overlap` can be dispatched on following classes:

- `top_rows_overlap,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `top_rows_overlap,matrix-method`, `matrix-class` class method
- `top_rows_overlap,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method

Examples

```
# no example
NULL
```

`top_rows_overlap-HierarchicalPartition-method`

Overlap of top rows on different nodes

Description

Overlap of top rows on different nodes

Usage

```
## S4 method for signature 'HierarchicalPartition'
top_rows_overlap(object, method = c("euler", "upset", "venn"), fill = NULL, ...)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`method`	`euler`: plot Euler diagram by `euler`; `upset`: draw the Upset plot by `UpSet`; `venn`: plot Venn diagram by `venn`; `correspondance`: use `correspond_between_rankings`.
`fill`	Filled color for the Euler diagram. The value should be a color vector. Transparency of 0.5 are added internally.
`...`	Additional arguments passed to `plot.euler`, `UpSet` or `correspond_between_rankings`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`top_elements_overlap`

Examples

```
data(golub_cola_rh)
top_rows_overlap(golub_cola_rh, method = "euler")
top_rows_overlap(golub_cola_rh, method = "upset")
top_rows_overlap(golub_cola_rh, method = "venn")
```

---

top_rows_overlap-matrix-method

Overlap of top rows from different top-value methods

Description

Overlap of top rows from different top-value methods

Usage

```
## S4 method for signature 'matrix'
top_rows_overlap(object, top_value_method = all_top_value_methods(),
    top_n = round(0.25*nrow(object)),
    method = c("euler", "upset", "venn", "correspondance"),
    fill = NULL, ...)
```

Arguments

`object`	A numeric matrix.
`top_value_method`	Methods defined in `all_top_value_methods`.
`top_n`	Number of top rows.
`method`	`euler`: plot Euler diagram by `euler`; `upset`: draw the Upset plot by `UpSet`; `venn`: plot Venn diagram by `venn`; `correspondance`: use `correspond_between_rankings`.
`fill`	Filled color for the Euler diagram. The value should be a color vector. Transparency of 0.5 are added internally.
`...`	Additional arguments passed to `plot.euler` or `correspond_between_rankings`.

Details

It first calculates scores for every top-value method and make plot by `top_elements_overlap`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`top_elements_overlap`

Examples

```
set.seed(123)
mat = matrix(rnorm(1000), nrow = 100)
top_rows_overlap(mat, top_n = 25)
```

[.ConsensusPartitionList

Subset a ConsensusPartitionList object

Description

Subset a ConsensusPartitionList object

Usage

```
## S3 method for class 'ConsensusPartitionList'
x[i, j, drop = TRUE]
```

Arguments

`x`	A `ConsensusPartitionList-class` object.
`i`	Index for top-value methods, character or nummeric.
`j`	Index for partitioning methods, character or nummeric.
`drop`	Whether drop class

Details

For a specific combination of top-value method and partitioning method, you can also subset by e.g. `x['SD:hclust']`.

Value

A `ConsensusPartitionList-class` object or a `ConsensusPartition-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
golub_cola[c("SD", "MAD"), c("hclust", "kmeans")]
golub_cola["SD", "kmeans"] # a ConsensusPartition object
golub_cola["SD:kmeans"] # a ConsensusPartition object
golub_cola[["SD:kmeans"]] # a ConsensusPartition object
golub_cola["SD", "kmeans", drop = FALSE] # still a ConsensusPartitionList object
golub_cola["SD:kmeans", drop = FALSE] # still a ConsensusPartitionList object
```

```
golub_cola["SD", ]
golub_cola[, "hclust"]
golub_cola[1:2, 1:2]
```

`[.HierarchicalPartition`

Subset the HierarchicalPartition object

Description

Subset the HierarchicalPartition object

Usage

```
## S3 method for class 'HierarchicalPartition'
x[i]
```

Arguments

`x`	A `HierarchicalPartition-class` object.
`i`	Index. The value should be numeric or a node ID.

Details

On each node, there is a `ConsensusPartition-class` object.

Note you cannot get a sub-hierarchy of the partition.

Value

A `ConsensusPartition-class` object.

Examples

```
data(golub_cola_rh)
golub_cola_rh["01"]
```

`[[.ConsensusPartitionList`

Subset a ConsensusPartitionList object

Description

Subset a ConsensusPartitionList object

Usage

```
## S3 method for class 'ConsensusPartitionList'
x[[i]]
```

Arguments

`x`	A `ConsensusPartitionList-class` object.
`i`	Character index for combination of top-value methods and partitioning method in a form of e.g. `SD:kmeans`.

Value

A `ConsensusPartition-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(golub_cola)
golub_cola[["SD:kmeans"]]
```

`[[.HierarchicalPartition`

Subset the HierarchicalPartition object

Description

Subset the HierarchicalPartition object

Usage

```
## S3 method for class 'HierarchicalPartition'
x[[i]]
```

Arguments

`x`	A `HierarchicalPartition-class` object
`i`	Index. The value should be numeric or a node ID.

Details

On each node, there is a `ConsensusPartition-class` object.

Note you cannot get a sub-hierarchy of the partition.

Value

A `ConsensusPartition-class` object.

Examples

```
# There is no example
NULL
```

Index