Package ‘cola’

October 16, 2019

Type Package

Title A Framework for Consensus and Hierarchical Partitioning

Version 1.0.1

Date 2019-09-16

Author Zuguang Gu

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

Depends R (>= 3.6.0)

Imports grDevices, graphics, grid, stats, utils, ComplexHeatmap (>= 2.0.0), matrixStats, GetoptLong, circlize (>= 0.4.7), GlobalOptions (>= 0.1.0), clue, parallel, RColorBrewer, cluster, skmeans, png, mclust, crayon, methods, xml2, microbenchmark, httr, knitr, markdown, digest, impute, brew, Rcpp (>= 0.11.0), BiocGenerics, eulerr

Suggests genefilter, mvtnorm, testthat (>= 0.3), data.tree, dendextend, samr, pamr, kohonen, NMF, WGCNA, Rtsne, umap, clusterProfiler, AnnotationDbi, gplots

Description Subgroup classification is a basic task in genomic data analysis, especially for gene expression data and methylation data. It can predict novel subgroups when there is nothing known about the data or it can test consistency between predicted subgroups with known annotations. The cola package provides a general framework for subgroup classification by consensus clustering. It has following features: 1. It modularizes the consensus clustering 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 compare results in a straightforward way. 4. It provides a new method to extract features which are more efficient to separate subgroups. 5. It allows doing partitioning in a hierarchical way to detect subgroups with relatively smaller difference. 6. It generates detailed reports for the complete analysis.

URL `https://github.com/jokergoo/cola`

VignetteBuilder knitr

biocViews Clustering, GeneExpression, Classification, Software

License MIT + file LICENSE

LinkingTo Rcpp

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

git_branch RELEASE_3_9

git_last_commit db87ed6

git_last_commit_date 2019-09-16

Date/Publication 2019-10-15

R topics documented:

`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)
```

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.

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

```
m = matrix(rnorm(200), 10)
rownames(m) = letters[1:10]
m[1, 1] = 1000
range(m)
m2 = adjust_matrix(m)
range(m2)
```

`adjust_outlier` *Adjust outliers*

Description

Adjust outliers

Usage

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

Arguments

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

Details

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

Value

A numeric vector with same length as the original one.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
x = rnorm(10)
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, depth = max_depth(object))
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`depth`	Depth in the hierarchy.

Value

A vector of node ID.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rh)
all_leaves(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, depth = max_depth(object))
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`depth`	Depth in the hierarchy.

Value

A vector of node ID.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rh)
all_nodes(cola_rh)
```

`all_partition_methods` *All supported partition methods*

Description

All supported partition methods

Usage

```
all_partition_methods()
```

Details

New partition methods can be registered by `register_partition_methods`.

Value

A vector of supported partition 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(cola_rl)
aPAC(get_consensus(cola_rl[1, 1], k = 2))
aPAC(get_consensus(cola_rl[1, 1], k = 3))
aPAC(get_consensus(cola_rl[1, 1], k = 4))
aPAC(get_consensus(cola_rl[1, 1], k = 5))
aPAC(get_consensus(cola_rl[1, 1], k = 6))
```

`ATC` *Ability to correlate other rows in the matrix*

Description

Ability to correlate other rows in the matrix

Usage

```
ATC(mat, cor_fun = stat::cor, min_cor = 0, power = 1,
    mc.cores = 1, n_sampling = 1000, q_sd = 0, ...)
```

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.
`mc.cores`	Mumber 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.
`q_sd`	Percentile of the standard deviation for the rows. Rows with values less than it are ignored.
`...`	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 the 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)
)
```

Value

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

Author(s)

Zuguang Gu <z.gu@dkfz.de>

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)))
```

`cola` *A bottle of cola*

Description

A bottle of cola

Usage

```
cola()
```

Details

Simply provide 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`	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`.
`fdr_cutoff`	Used in `get_signatures,ConsensusPartition-method`.

Examples

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

`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(), env = parent.frame())
```

Arguments

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

Details

It generates report for a specific combination of top-value method and partition 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, env = parent.frame())
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`output_dir`	The output directory where the report is put.
`mc.cores`	Multiple cores to use.
`env`	Where The objects in the report are found, internally used.

Details

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

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
## Not run:
data(cola_rl)
cola_report(cola_rl[c("sd", "MAD"), c("hclust", "skmeans")], output_dir = "~/test_cola_cl_report")

## End(Not run)
```

`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,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `cola_report,ConsensusPartition-method`, `ConsensusPartition-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, mc.cores = 1, 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.
`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.

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

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
## Not run:
data(cola_rh)
cola_report(cola_rh, output_dir = "~/test_cola_rh_report")

## End(Not run)
```

`cola_rh` *Example HierarchicalPartition object*

Description

Example HierarchicalPartition object

Usage

```
data(cola_rh)
```

Details

Following code was used to generate `cola_rh`:

```
  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)
  cola_rh = hierarchical_partition(m, top_n = c(20, 30, 40))
```

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rh)
cola_rh
```

`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, top_n = c(20, 30, 40))
```

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rl)
cola_rl
```

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

Description

Collect classes from ConsensusPartition object

Usage

```
## S4 method for signature 'ConsensusPartition'
collect_classes(object, internal = FALSE, show_row_names = FALSE,
    anno = get_anno(object), anno_col = get_anno_col(object))
```

Arguments

`object`	A `ConsensusPartition-class` object.
`internal`	Used internally.
`show_row_names`	Whether show row names in the heatmap (which is the column name in the original matrix).
`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 class IDs 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(cola_rl)
collect_classes(cola_rl["sd", "kmeans"])
```

`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,
    anno = get_anno(object), anno_col = get_anno_col(object))
```

Arguments

`object`	A `ConsensusPartitionList-class` object returned by `run_all_consensus_partition_methods`.
`k`	Number of partitions.
`show_column_names`	Whether show column names in the heatmap (which is the column name in the original matrix).
`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`.

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(cola_rl)
collect_classes(cola_rl, 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, depth = max_depth(object),
    anno = get_anno(object[1]), anno_col = get_anno_col(object[1]))
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`depth`	Depth of the hierarchy.
`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`.

Details

The function plots the hierarchy of the classes.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rh)
collect_classes(cola_rh)
collect_classes(cola_rh, 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

```
## Not run:
data(cola_rl)
collect_plots(cola_rl["sd", "kmeans"])

## End(Not run)
```

`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, ...)
```

Arguments

`object`	A `ConsensusPartitionList-class` object from `run_all_consensus_partition_methods`.
`k`	Number of partitions.
`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 partition methods.
`verbose`	Whether to print message.
`mc.cores`	Number of cores. On OSX it is enforced to be 1.
`...`	other Arguments passed to corresponding `fun`.

Details

Plots for all combinations of top-value methods and parittion 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(cola_rl)
collect_plots(cola_rl, k = 3)
## Not run:
collect_plots(cola_rl, k = 3, fun = membership_heatmap)
collect_plots(cola_rl, k = 3, fun = get_signatures)

## End(Not run)
```

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

Examples

```
# no example
NULL
```

`collect_plots-HierarchicalPartition-method`
Collect plots from HierarchicalPartition object

Description

Collect plots from HierarchicalPartition object

Usage

```
## S4 method for signature 'HierarchicalPartition'
collect_plots(object, depth = max_depth(object),
    fun = consensus_heatmap, verbose = TRUE, mc.cores = 1, heatmap_size = "2cm",
    add_labels = TRUE, ...)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`depth`	Depth in the hierarchy.
`fun`	Function used to generate plots. Valid functions are `consensus_heatmap`, `plot_ecdf`, `membership_heatmap`, `get_signatures` and `dimension_reduction`.
`verbose`	Whether to print message.
`mc.cores`	Number of cores. On OSX it is enforced to be 1.
`heatmap_size`	Size of the heatmaps on the plot. The value should be in e.g. "2cm", "10mm" or "0.5inche".
`add_labels`	Whether add node IDs.
`...`	other Arguments passed to corresponding `fun`.

Details

The hierarchy represents as a circular dendrogram where plots are on the nodes.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rh)
collect_plots(cola_rh)
## Not run:
collect_plots(cola_rh, fun = membership_heatmap)
collect_plots(cola_rh, fun = get_signatures)

## End(Not run)
```

`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 partitions
`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 partitions.

Examples

```
data(cola_rl)
collect_stats(cola_rl, 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,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `colnames,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `colnames,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# 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_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, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of partitions. Value should be a vector.
`...`	Other arguments passed to `get_signatures,ConsensusPartition-method`.

Details

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

Examples

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

`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.
`class`	Consensus class IDs.

Details

Note class IDs in `membership_each` should already be adjusted to the consensus class IDs to let `sum(x_single == x_consensus)` reach maximum.

The concordance score is the mean proportion of samples having the same class ID as the consensus class ID among runs.

This function is used internally.

Value

A numeric value.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

`ConsensusPartition-class`

The ConsensusPartition class

Description

The ConsensusPartition class

Methods

The `ConsensusPartition-class` has following methods:

- `consensus_partition`**:** constructor method, run consensus partition with a specified top-value method and a partition method.
- `select_partition_number,ConsensusPartition-method`**:** make a list of plots to select optimized number of partitions.
- `consensus_heatmap,ConsensusPartition-method`**:** make heatmap of the consensus matrix.
- `membership_heatmap,ConsensusPartition-method`**:** make heatmap of the membership in every random sampling.
- `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 partitions.
- `collect_classes,ConsensusPartition-method`**:** make heatmap of classes with different numbers of partitions.
- `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 in random samplings.
- `get_stats,ConsensusPartition-method`**:** get metrics for the consensus clustering.
- `get_classes,ConsensusPartition-method`**:** get the consensus class IDs and other columns.
- `suggest_best_k,ConsensusPartition-method`**:** guess the best number of partitions.
- `test_to_known_factors,ConsensusPartition-method`**:** test correlation between predicted classes and known factors, if available.
- `cola_report,ConsensusPartition-method`**:** generate a HTML report for the whole analysis.
- `GO_enrichment,ConsensusPartition-method`**:** perform GO 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 partition 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 class IDs merging from all methods.
- `get_matrix,ConsensusPartition-method`**:** get the original matrix.
- `get_stats,ConsensusPartitionList-method`**:** get metrics 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 partitions for all methods.
- `collect_plots,ConsensusPartitionList-method`**:** collect plots from all combinations of top-value methods and partition methods with choosing a plotting function.
- `collect_classes,ConsensusPartitionList-method`**:** make a plot which contains predicted classes from all combinations of top-value methods and partition methods.
- `test_to_known_factors,ConsensusPartitionList-method`**:** test correlation between predicted classes and known annotations, if provided.
- `cola_report,ConsensusPartitionList-method`**:** generate a HTML report for the whole analysis.
- `GO_enrichment,ConsensusPartitionList-method`**:** perform GO 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 for the consensus matrix

Description

Heatmap for the consensus matrix

Usage

```
## S4 method for signature 'ConsensusPartition'
consensus_heatmap(object, k, internal = FALSE,
    anno = get_anno(object), anno_col = get_anno_col(object),
    show_row_names = FALSE, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of partitions.
`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)
`...`	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 classes.
- consensus matrix.
- more annotations if provided as `anno`

One thing that is very important to note is that since we already know the consensus classes 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(cola_rl)
consensus_heatmap(cola_rl["sd", "hclust"], k = 3)
```

`consensus_partition` *Consensus partition*

Description

Consensus partition

Usage

```
consensus_partition(data,
    top_value_method = "ATC",
    top_n = seq(min(1000, round(nrow(data)*0.1)),
    min(5000, round(nrow(data)*0.5)),
    length.out = 5),
    partition_method = "skmeans",
    max_k = 6,
    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,
    .env = NULL)
```

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 partition method. Available methods are in `all_partition_methods`. Use `register_partition_methods` to add a new partition method.
`max_k`	Maximal number of partitions to try. The function will try `2:max_k` partitions.
`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 partition 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.
`.env`	An environment, internally used.

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 partitions and calculate consensus partitions.

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 partition with multiple top-value methods and multiple partition methods. `hierarchical_partition` runs consensus partition hierarchically.

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)
cp = consensus_partition(m, partition_repeat = 10, top_n = c(10, 20, 50))
cp
```

`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 = brewer_pal_set1_col[1:length(lt)], ...)
```

Arguments

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

Details

It makes plots for every pairwise comparisons 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, vc = 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 correspondance.
`transparency`	Transparency of the connection 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 is 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. The top n elements under corresponding metric are highlighted by vertical color lines in both panels. The left and right panels also show 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>

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)
```

`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.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(),
    internal = FALSE,
    silhouette_cutoff = 0.5, remove = FALSE,
    scale_rows = TRUE, verbose = TRUE, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of partitions.
`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`.
`control`	A list of parameters for `Rtsne` or `umap`.
`internal`	Internally 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 perform scaling on matrix rows.
`verbose`	Whether print messages.
`...`	Other arguments.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rl)
dimension_reduction(cola_rl["sd", "kmeans"], 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,HierarchicalPartition-method`, `HierarchicalPartition-class` class method
- `dimension_reduction,matrix-method`, `matrix-class` class method
- `dimension_reduction,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

---

dimension_reduction-HierarchicalPartition-method
Visualize columns after dimension reduction

Description

Visualize columns after dimension reduction

Usage

```
## S4 method for signature 'HierarchicalPartition'
dimension_reduction(object,
    depth = max_depth(object), parent_node,
    top_n = NULL, method = c("PCA", "MDS", "t-SNE", "UMAP"),
    silhouette_cutoff = 0.5, scale_rows = TRUE)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`depth`	Depth of the hierarchy.
`top_n`	Top n rows to use. By default it uses all rows in the original matrix.
`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`.
`silhouette_cutoff`	Cutoff of silhouette score. Data points with values less than it will be mapped to small points.
`scale_rows`	Whether perform scaling on matrix rows.

Details

The class IDs are extract at `depth`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rh)
dimension_reduction(cola_rh)
dimension_reduction(cola_rh, parent_node = "00")
```

`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 = "",
    method = c("PCA", "MDS", "t-SNE", "UMAP"),
    pc = 1:2, control = list(),
    scale_rows = TRUE,
    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.
`internal`	Internally used.
`verbose`	Whether print messages.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

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(cola_rl)
FCC(get_consensus(cola_rl[1, 1], k = 2))
FCC(get_consensus(cola_rl[1, 1], k = 3))
FCC(get_consensus(cola_rl[1, 1], k = 4))
FCC(get_consensus(cola_rl[1, 1], k = 5))
FCC(get_consensus(cola_rl[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 determination of the best k.

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

Examples

```
# no example
NULL
```

`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_classes-ConsensusPartition-method`

Get class IDs from the ConsensusPartition object

Description

Get class IDs from the ConsensusPartition object

Usage

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

Arguments

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

Value

A data frame with class IDs 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 class IDs from every k.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rl)
obj = cola_rl["sd", "kmeans"]
get_classes(obj, k = 2)
get_classes(obj)
```

`get_classes-ConsensusPartitionList-method`

Get class IDs from the ConsensusPartitionList object

Description

Get class IDs from the ConsensusPartitionList object

Usage

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

Arguments

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

Details

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

Value

A data frame with class IDs 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(cola_rl)
get_classes(cola_rl, 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

Examples

```
# no example
NULL
```

---

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, depth = max_depth(object))
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`depth`	Depth of the hierarchy.

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(cola_rh)
get_classes(cola_rh)
get_classes(cola_rh, depth = 2)
```

---

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 partitions.

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(cola_rl)
obj = cola_rl["sd", "kmeans"]
get_consensus(obj, k = 2)
```

`get_hierarchy` *Get the partition hierarchy*

Description

Get the partition hierarchy

Usage

```
get_hierarchy(object, depth = max_depth(object))
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`depth`	Depth of the hierarchy.

Value

A `dendrogram` object.

Examples

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

`get_matrix-ConsensusPartition-method`

Get the original matrix

Description

Get the original matrix

Usage

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

Arguments

`object`	A `ConsensusPartition-class` object

Value

A numeric matrix.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rl)
obj = cola_rl["sd", "kmeans"]
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(cola_rl)
get_matrix(cola_rl)
```

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

Examples

```
# 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 partitions.
`each`	Whether return the percentage membership matrix which is summarized from all partitions or the individual membership in every random partition.

Details

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

The percent membership matrix is calculated by `cl_consensus`.

Value

If `each` == `TRUE`, it returns 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,ConsensusPartitionList-method` summarizes membership from partitions from all combinations of top-value methods and partition methods.

Examples

```
data(cola_rl)
obj = cola_rl["sd", "kmeans"]
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 partitions.

Details

The membership matrix (the probability of each sample to be in one group, 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 instable 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 partition method.

Examples

```
data(cola_rl)
get_membership(cola_rl, 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,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `get_membership,ConsensusPartition-method`, `ConsensusPartition-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 partitions.
`unique`	Whether 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(cola_rl)
obj = cola_rl["sd", "kmeans"]
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,
    silhouette_cutoff = 0.5,
    fdr_cutoff = cola_opt$fdr_cutoff,
    group_diff = cola_opt$group_diff,
    scale_rows = object@scale_rows,
    row_km = NULL,
    diff_method = c("Ftest", "ttest", "samr", "pamr", "one_vs_others"),
    anno = get_anno(object),
    anno_col = get_anno_col(object),
    internal = FALSE,
    show_row_dend = FALSE,
    show_column_names = FALSE, use_raster = TRUE,
    plot = TRUE, verbose = TRUE, seed = 888,
    left_annotation = NULL, right_annotation = NULL,
   col = if(scale_rows) c("green", "white", "red") else c("blue", "white", "red"),
    ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of partitions.
`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.
`group_diff`	Cutoff for the maximal difference between group means.
`scale_rows`	Whether apply row scaling when making the heatmap.

`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.
`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`.
`col`	Colors.
`...`	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).

`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 expression (depending rows are scaled or not) in each subgroup.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

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

Examples

```
# no example
NULL
```

`get_signatures-HierarchicalPartition-method`
Get signatures rows

Description

Get signatures rows

Usage

```
## S4 method for signature 'HierarchicalPartition'
get_signatures(object, depth = max_depth(object),
    scale_rows = object[1]@scale_rows,
    anno = get_anno(object),
    anno_col = get_anno_col(object),
    show_column_names = FALSE,
    verbose = TRUE, plot = TRUE,
    silhouette_cutoff = 0.5,
    ...)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`depth`	Depth of the hierarchy.
`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.
`verbose`	Whether to print messages.
`plot`	Whether to make the plot.
`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`.
`...`	Other arguments

Details

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

Value

A list of row indices where rows are significantly different between subgroups in at least one node. Other columns in the returned data frames are whether the rows are significantly different in the node.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
## Not run:
data(cola_rh)
get_signatures(cola_rh)

## End(Not run)
```

---

get_stats-ConsensusPartition-method

Get statistics for the consensus partition

Description

Get statistics for the consensus partition

Usage

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

Arguments

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

Details

The statistics are:

PAC 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 the Rand index which 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(cola_rl)
obj = cola_rl["sd", "kmeans"]
get_stats(obj)
get_stats(obj, k = 2)
```

`get_stats-ConsensusPartitionList-method`

Get statistics for consensus partitions from all methods

Description

Get statistics for consensus partitions from all methods

Usage

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

Arguments

`object`	A `ConsensusPartitionList-class` object.
`k`	Number of partitions. 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 partition methods.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rl)
get_stats(cola_rl, 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
```

`GO_enrichment-ANY-method` *Perform Gene Ontology Enrichment on Signature Genes*

Description

Perform Gene Ontology Enrichment on Signature Genes

Usage

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

Arguments

`object`	A vector of gene IDs.
`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`	"BP": biological processes, "MF": molecular functions, "CC": cellular components.

`min_set_size`	The minimal size of the GO gene sets.
`max_set_size`	The maximal size of the GO gene sets.
`verbose`	Whether to print messages.
`...`	Other arguments.

Value

A list of three data frames which correspond to results for three GO catalogues:

- `BP`: biological processes
- `MF`: molecular functions
- `CC`: cellular components

Examples

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

GO_enrichment-ConsensusPartition-method

Perform Gene Ontology Enrichment on Signature Genes

Description

Perform Gene Ontology Enrichment on Signature Genes

Usage

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

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`	"BP": biological processes, "MF": molecular functions, "CC": cellular components.
`min_set_size`	The minimal size of the GO gene sets.
`max_set_size`	The maximal size of the GO gene sets.
`verbose`	Whether to print messages.
`...`	Other arguments.

Value

A list of three data frames which correspond to results for three GO catalogues:

- `BP`: biological processes
- `MF`: molecular functions
- `CC`: cellular components

Examples

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

`GO_enrichment-ConsensusPartitionList-method`

Perform Gene Ontology Enrichment on Signature Genes

Description

Perform Gene Ontology Enrichment on Signature Genes

Usage

```
## S4 method for signature 'ConsensusPartitionList'
GO_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 = c("BP", "MF", "CC"),
    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`	"BP": biological processes, "MF": molecular functions, "CC": cellular components.
`min_set_size`	The minimal size of the GO gene sets.
`max_set_size`	The maximal size of the GO gene sets.

Details

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

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

Value

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

Examples

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

`GO_enrichment-dispatch`

Method dispatch page for GO_enrichment

Description

Method dispatch page for `GO_enrichment`.

Dispatch

`GO_enrichment` can be dispatched on following classes:

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

Examples

```
# no example
NULL
```

`GO_enrichment-HierarchicalPartition-method`

Perform Gene Ontology Enrichment on Signature Genes

Description

Perform Gene Ontology Enrichment on Signature Genes

Usage

```
## S4 method for signature 'HierarchicalPartition'
GO_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",
    min_set_size = 10, max_set_size = 1000)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`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.
`min_set_size`	The minimal size of the GO gene sets.
`max_set_size`	The maximal size of the GO gene sets.

Details

On each node of the partition hierarchy, the signature genes are extracted based on the best k.

It calls `GO_enrichment,ConsensusPartition-method` on the consensus partitioning results on each node.

Value

A list where each element in the list corresponds to enrichment results on a node.

Examples

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

`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.

`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.

`GO_enrichment,HierarchicalPartition-method`: perform GO 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
```

`hierarchical_partition` *Hierarchical partition*

Description

Hierarchical partition

Usage

```
hierarchical_partition(data, top_value_method = "MAD", partition_method = "kmeans",
    PAC_cutoff = 0.1, silhouette_cutoff = 0.5,
    min_samples = 6, min_signatures = c(50, 0.05), max_k = 4, verbose = TRUE,
    mc.cores = 1, ...)
```

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`.
`partition_method`	A single partition method. Available methods are in `all_partition_methods`.
`PAC_cutoff`	The cutoff of PAC scores to determine whether to continue looking for subgroups.
`silhouette_cutoff`	Cutoff for silhouette scores.
`min_samples`	The cutoff of number of samples to determine whether to continue looking for subgroups.
`min_signatures`	Minimal number of signatures to determine whether to continue looking for subgroups. The value should be a vector of length two. The first value is the absolute number of signatures and the second value is the proportion of signatures in the matrix. The hierarchical partitioning stops when both criterions are met.
`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.
`verbose`	Whether print message.
`mc.cores`	Multiple cores to use.
`...`	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)
cola_rh = hierarchical_partition(m, top_n = c(20, 30, 40), PAC_cutoff = 0.3)

## End(Not run)
data(cola_rh)
cola_rh
```

`knitr_add_tab_item` *Add one JavaScript tab in the report*

Description

Add one 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 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 callings 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, randomly picked 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 `GO_enrichment`.

Examples

```
## Not run:
    map_to_entrez_id("ENSEMBL")

## End(Not run)
```

`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(cola_rh)
max_depth(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 = get_anno(object), anno_col = get_anno_col(object),
    show_column_names = FALSE, ...)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`k`	Number of partitions.
`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).
`...`	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(cola_rl)
membership_heatmap(cola_rl["sd", "hclust"], k = 3)
```

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

Examples

```
# 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
```

---

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`	class IDs. If it is provided, samples with silhouette score less than 5th percential are removed.

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

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

Description

Plot the empirical cumulative distribution curve (ECDF) 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(cola_rl)
plot_ecdf(cola_rl["sd", "hclust"])
```

`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 statistics slot in the ConsensusPartitionList object, used internally.

Examples

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

`register_NMF` *Register NMF partition method*

Description

Register NMF partition method

Usage

```
register_NMF()
```

Details

NMF analysis is performed by `nmf`.

Examples

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

`register_partition_methods` *Register user-defined partition functions*

Description

Register user-defined partition functions

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 wil be no row scaling when doing partition. 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 partitions. 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 of the R functions apply on rows and some of the 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 partition methods will be used as defaults in `run_all_consensus_partition_methods`.

To remove a partition method, use `remove_partition_methods`.

There are following default partition 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 partition 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 other two pre-defined partition 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 partition 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 partition method*

Description

Register SOM partition 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(...)
```

Arguments

`...`	A named list of 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 IDs according to the reference ID*

Description

Relabel class IDs according to the reference ID

Usage

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

Arguments

`class`	A vector of class IDs.
`ref`	A vector of reference IDs.
`full_set`	The full set of ID levels.
`return_map`	Whether return the mapping or the adjusted labels.

Details

In partition, the exact value of the class ID 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. Here `relabel_class` function switches the labels in `class` vector according to the labels in `ref` vector to maximize `sum(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 IDs in `class` and values correspond to `ref`, which means `map = relabel_class(class,ref); map[class]` returns the relabelled IDs.

The returned object attaches a data frame with three columns:

- original IDs in `class`
- adjusted IDs according to `ref`
- reference IDs in `ref`

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

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)
```

`remove_partition_methods`

Remove partition methods

Description

Remove partition methods

Usage

```
remove_partition_methods(method)
```

Arguments

`method`	Name of the partition 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,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `rownames,ConsensusPartition-method`, `ConsensusPartition-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 partition for all combinations of methods

Description

Consensus partition 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,
    top_n = seq(min(1000, round(nrow(data)*0.1)),
    min(5000, round(nrow(data)*0.5)),
    length.out = 5),
    mc.cores = 1, anno = NULL, anno_col = NULL,
    sample_by = "row", p_sampling = 0.8, partition_repeat = 50,
    scale_rows = NULL, verbose = TRUE)
```

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 do partition on samples. Allowed methods are in `all_partition_methods` and can be self-added by `register_partition_methods`.
`max_k`	Maximal number of partitions to try. The function will try `2:max_k` partitions.
`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.
`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`	Ahether to print messages.

Details

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

It also adjsuts the class IDs 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)
data(cola_rl)
cola_rl
```

`select_partition_number-ConsensusPartition-method`

Several plots for determining the optimized number of partitions

Description

Several plots for determining the optimized number of partitions

Usage

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

Arguments

`object`	A `ConsensusPartition-class` object.
`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(cola_rl)
select_partition_number(cola_rl["sd", "hclust"])
```

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

Examples

```
# no example
NULL
```

`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(cola_rh)
cola_rh
```

`submit_to_david` *Perform DAVID analysis*

Description

Perform DAVID analysis

Usage

```
submit_to_david(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

https://david.ncifcrf.gov

Examples

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

---

suggest_best_k-ConsensusPartition-method

Suggest the best number of partitions

Description

Suggest the best number of partitions

Usage

```
## S4 method for signature 'ConsensusPartition'
suggest_best_k(object, jaccard_index_cutoff = 0.95)
```

Arguments

`object`	A `ConsensusPartition-class` object.
`jaccard_index_cutoff`	The cutoff for Jaccard index compared to previous k.

Details

The best k is selected according to following rules:

1. k with rand index larger than `jaccard_index_cutoff` are removed. If all k are removed, the best k is defined as `NA`. 2. If there are some k having `1-PAC` larger than 0.9, the largest k is selected as the best k. 3. If it does not fit rule 2, the k with highest vote of highest 1-PAC, mean_silhouette and concordance scores is selected as the best k.

`suggest_best_k` function only gives suggestion of selecting a reasonable best k. Users still need to look at the plots (e.g. by `select_partition_number` or `consensus_heatmap` functions), or even by checking whether the subgrouping gives a reasonable signatures by `get_signatures`, to pick a reasonable k that best explains their study.

The best k with 1-PAC larger than 0.9 is treated as a stable partition.

Value

The best k.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rl)
obj = cola_rl["sd", "kmeans"]
suggest_best_k(obj)
```

---

suggest_best_k-ConsensusPartitionList-method

Suggest the best number of partitions

Description

Suggest the best number of partitions

Usage

```
## S4 method for signature 'ConsensusPartitionList'
suggest_best_k(object, jaccard_index_cutoff = 0.95)
```

Arguments

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

Details

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

1-PAC score higher than 0.95 is treated as very stable partition and higher than 0.9 is treated as stable partition.

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(cola_rl)
suggest_best_k(cola_rl)
```

`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` *Suggest the best number of partitions*

Description

Suggest 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(cola_rh)
suggest_best_k(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;
- 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 classes and known factors

---

Description

Test correspondance between predicted classes 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 partitions. 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 it are omit.
`verbose`	Whether to print messages.

Value

A data frame with columns:

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

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`test_between_factors`

Examples

```
data(cola_rl)
test_to_known_factors(cola_rl[1, 1], known = 1:40)
```

`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 partitions. 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,ConsensusPartit`
and merges results afterwards.

Value

A data frame with columns:

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

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`test_between_factors`, `test_to_known_factors,ConsensusPartition-method`

Examples

```
data(cola_rl)
test_to_known_factors(cola_rl, known = 1:40)
```

`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,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method
- `test_to_known_factors,ConsensusPartition-method`, `ConsensusPartition-class` class method

Examples

```
# no example
NULL
```

---

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]),
    depth = 2:max_depth(object), verbose = FALSE)
```

Arguments

`object`	A `HierarchicalPartition-class` object.
`depth`	Depth of the hierarchy.
`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

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rh)
test_to_known_factors(cola_rh, known = 1:60)
```

`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", "venn", "correspondance"), ...)
```

Arguments

`object`	A list which contains values from different metrics.
`top_n`	Number of top rows.
`method`	`euler`: plot Euler diagram by `euler`; `venn`: plot Venn diagram by `venn`; `correspondance`: use `correspond_between_rankings`.
`...`	Additional arguments passed to `plot.euler` 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 = 25, method = "venn")
top_elements_overlap(lt, top_n = 25, method = "correspondance")
```

---

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

Examples

```
# 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 scale rows.

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", "venn", "correspondance"), ...)
```

Arguments

`object`	A `ConsensusPartitionList-class` object.
`top_n`	Number of top rows.
`method`	`euler`: plot Euler diagram by `euler`; `venn`: plot Venn diagram by `venn`; `correspondance`: use `correspond_between_rankings`.
`...`	Additional arguments passed to `plot.euler` or `correspond_between_rankings`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`top_elements_overlap`

Examples

```
data(cola_rl)
top_rows_overlap(cola_rl, method = "venn")
top_rows_overlap(cola_rl, 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,matrix-method`, `matrix-class` class method
- `top_rows_overlap,ConsensusPartitionList-method`, `ConsensusPartitionList-class` class method

Examples

```
# no example
NULL
```

---

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", "venn", "correspondance"), ...)
```

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`; `venn`: plot Venn diagram by `venn`; `correspondance`: use `correspond_between_rankings`.
`...`	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 partition methods, character or nummeric.
`drop`	Whether drop class

Details

For a specific combination of top-value method and partition 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(cola_rl)
cola_rl[c("sd", "MAD"), c("hclust", "kmeans")]
cola_rl["sd", "kmeans"] # a ConsensusPartition object
cola_rl["sd:kmeans"] # a ConsensusPartition object
cola_rl[["sd:kmeans"]] # a ConsensusPartition object
cola_rl["sd", "kmeans", drop = FALSE] # still a ConsensusPartitionList object
cola_rl["sd:kmeans", drop = FALSE] # still a ConsensusPartitionList object
cola_rl["sd", ]
cola_rl[, "hclust"]
cola_rl[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(cola_rh)
cola_rh["01"]
cola_rh[2]
```

[[.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 partition method in a form of e.g. `sd:MAD`.

Value

A `ConsensusPartition-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
data(cola_rl)
cola_rl[["sd:MAD"]]
```

[[.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(cola_rh)
cola_rh[["01"]]
cola_rh[[2]]
```

Index