

# Package: cola (via r-universe)

June 17, 2026

**Type** Package

**Title** A Framework for Consensus Partitioning

**Version** 2.18.0

**Date** 2026-01-30

**Depends** R (>= 4.0.0)

**Imports** grDevices, graphics, grid, stats, utils, ComplexHeatmap (>= 2.5.4), matrixStats (>= 1.2.0), GetoptLong, circlize (>= 0.4.7), GlobalOptions (>= 0.1.0), clue, parallel, RColorBrewer, cluster, skmeans, png, mclust, crayon, methods, xml2, microbenchmark, httr, knitr (>= 1.4.0), markdown (>= 1.6), digest, impute, brew, Rcpp (>= 0.11.0), BiocGenerics, eulerr, foreach, doParallel, doRNG, irlba

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

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

**URL** <https://github.com/jokergoo/cola>,  
[https://jokergoo.github.io/cola\\_collection/](https://jokergoo.github.io/cola_collection/)

**VignetteBuilder** knitr

**biocViews** Clustering, GeneExpression, Classification, Software

**License** MIT + file LICENSE

**LinkingTo** Rcpp

**Config/pak/sysreqs** libpng-dev libxml2-dev libssl-dev perl libclang-dev

**Repository** <https://bioc-release.r-universe.dev>

**Date/Publication** 2026-04-28 12:49:26 UTC

**RemoteUrl** <https://github.com/bioc/cola>

**RemoteRef** RELEASE\_3\_23

**RemoteSha** a4096e1e990a4b757effc1286bdf5d3b2bd44a48

## Contents

[.ConsensusPartitionList . . . . .	6
[.HierarchicalPartition . . . . .	7
[[.ConsensusPartitionList . . . . .	7
[[.HierarchicalPartition . . . . .	8
adjust_matrix . . . . .	9
adjust_outlier . . . . .	10
all_leaves-HierarchicalPartition-method . . . . .	11
all_nodes-HierarchicalPartition-method . . . . .	11
all_partition_methods . . . . .	12
all_top_value_methods . . . . .	13
aPAC . . . . .	13
ATC . . . . .	14
ATC_approx . . . . .	16
cola . . . . .	17
cola_opt . . . . .	17
cola_report-ConsensusPartition-method . . . . .	18
cola_report-ConsensusPartitionList-method . . . . .	19
cola_report-dispatch . . . . .	20
cola_report-HierarchicalPartition-method . . . . .	21
cola_rl . . . . .	22
collect_classes-ConsensusPartition-method . . . . .	22
collect_classes-ConsensusPartitionList-method . . . . .	23
collect_classes-dispatch . . . . .	25
collect_classes-HierarchicalPartition-method . . . . .	25
collect_plots-ConsensusPartition-method . . . . .	26
collect_plots-ConsensusPartitionList-method . . . . .	27
collect_plots-dispatch . . . . .	28
collect_stats-ConsensusPartition-method . . . . .	29
collect_stats-ConsensusPartitionList-method . . . . .	29

collect_stats-dispatch . . . . .	30
colnames-ConsensusPartition-method . . . . .	31
colnames-ConsensusPartitionList-method . . . . .	31
colnames-dispatch . . . . .	32
colnames-DownSamplingConsensusPartition-method . . . . .	32
colnames-HierarchicalPartition-method . . . . .	33
compare_partitions-ConsensusPartition-method . . . . .	33
compare_signatures-ConsensusPartition-method . . . . .	34
compare_signatures-dispatch . . . . .	35
compare_signatures-HierarchicalPartition-method . . . . .	36
concordance . . . . .	36
config_ATC . . . . .	37
consensus_heatmap-ConsensusPartition-method . . . . .	38
consensus_partition . . . . .	40
consensus_partition_by_down_sampling . . . . .	42
ConsensusPartition-class . . . . .	44
ConsensusPartitionList-class . . . . .	45
correspond_between_rankings . . . . .	46
correspond_between_two_rankings . . . . .	47
david_enrichment . . . . .	48
dim.ConsensusPartition . . . . .	50
dim.ConsensusPartitionList . . . . .	50
dim.DownSamplingConsensusPartition . . . . .	51
dim.HierarchicalPartition . . . . .	51
dimension_reduction-ConsensusPartition-method . . . . .	52
dimension_reduction-dispatch . . . . .	53
dimension_reduction-DownSamplingConsensusPartition-method . . . . .	53
dimension_reduction-HierarchicalPartition-method . . . . .	54
dimension_reduction-matrix-method . . . . .	55
DownSamplingConsensusPartition-class . . . . .	57
FCC . . . . .	57
find_best_km . . . . .	58
functional_enrichment-ANY-method . . . . .	59
functional_enrichment-ConsensusPartition-method . . . . .	60
functional_enrichment-ConsensusPartitionList-method . . . . .	61
functional_enrichment-dispatch . . . . .	62
functional_enrichment-HierarchicalPartition-method . . . . .	63
get_anno-ConsensusPartition-method . . . . .	64
get_anno-ConsensusPartitionList-method . . . . .	65
get_anno-dispatch . . . . .	65
get_anno-DownSamplingConsensusPartition-method . . . . .	66
get_anno-HierarchicalPartition-method . . . . .	67
get_anno_col-ConsensusPartition-method . . . . .	67
get_anno_col-ConsensusPartitionList-method . . . . .	68
get_anno_col-dispatch . . . . .	69
get_anno_col-HierarchicalPartition-method . . . . .	69
get_children_nodes-HierarchicalPartition-method . . . . .	70
get_classes-ConsensusPartition-method . . . . .	70

get_classes-ConsensusPartitionList-method . . . . .	71
get_classes-dispatch . . . . .	72
get_classes-DownSamplingConsensusPartition-method . . . . .	72
get_classes-HierarchicalPartition-method . . . . .	73
get_consensus-ConsensusPartition-method . . . . .	74
get_matrix-ConsensusPartition-method . . . . .	75
get_matrix-ConsensusPartitionList-method . . . . .	75
get_matrix-dispatch . . . . .	76
get_matrix-DownSamplingConsensusPartition-method . . . . .	77
get_matrix-HierarchicalPartition-method . . . . .	77
get_membership-ConsensusPartition-method . . . . .	78
get_membership-ConsensusPartitionList-method . . . . .	79
get_membership-dispatch . . . . .	80
get_param-ConsensusPartition-method . . . . .	80
get_signatures-ConsensusPartition-method . . . . .	81
get_signatures-dispatch . . . . .	84
get_signatures-DownSamplingConsensusPartition-method . . . . .	85
get_signatures-HierarchicalPartition-method . . . . .	85
get_stats-ConsensusPartition-method . . . . .	87
get_stats-ConsensusPartitionList-method . . . . .	88
get_stats-dispatch . . . . .	89
golub_cola . . . . .	89
golub_cola_ds . . . . .	91
golub_cola_rh . . . . .	91
hierarchical_partition . . . . .	92
HierarchicalPartition-class . . . . .	94
is_best_k-ConsensusPartition-method . . . . .	95
is_best_k-ConsensusPartitionList-method . . . . .	96
is_best_k-dispatch . . . . .	97
is_leaf_node-HierarchicalPartition-method . . . . .	97
is_stable_k-ConsensusPartition-method . . . . .	98
is_stable_k-ConsensusPartitionList-method . . . . .	99
is_stable_k-dispatch . . . . .	99
knee_finder2 . . . . .	100
knitr_add_tab_item . . . . .	100
knitr_insert_tabs . . . . .	101
map_to_entrez_id . . . . .	102
max_depth-HierarchicalPartition-method . . . . .	103
membership_heatmap-ConsensusPartition-method . . . . .	104
merge_node-HierarchicalPartition-method . . . . .	105
merge_node_param . . . . .	105
ncol-ConsensusPartition-method . . . . .	106
ncol-ConsensusPartitionList-method . . . . .	107
ncol-dispatch . . . . .	107
ncol-DownSamplingConsensusPartition-method . . . . .	108
ncol-HierarchicalPartition-method . . . . .	108
node_info-HierarchicalPartition-method . . . . .	109
node_level-HierarchicalPartition-method . . . . .	109

nrow-ConsensusPartition-method . . . . .	110
nrow-ConsensusPartitionList-method . . . . .	111
nrow-dispatch . . . . .	111
nrow-HierarchicalPartition-method . . . . .	112
PAC . . . . .	112
plot_ecdf-ConsensusPartition-method . . . . .	113
predict_classes-ConsensusPartition-method . . . . .	114
predict_classes-dispatch . . . . .	116
predict_classes-matrix-method . . . . .	117
print.hc_table_suggest_best_k . . . . .	119
recalc_stats . . . . .	119
register_NMF . . . . .	120
register_partition_methods . . . . .	120
register_SOM . . . . .	122
register_top_value_methods . . . . .	122
relabel_class . . . . .	124
remove_partition_methods . . . . .	125
remove_top_value_methods . . . . .	126
rownames-ConsensusPartition-method . . . . .	126
rownames-ConsensusPartitionList-method . . . . .	127
rownames-dispatch . . . . .	127
rownames-HierarchicalPartition-method . . . . .	128
run_all_consensus_partition_methods . . . . .	128
select_partition_number-ConsensusPartition-method . . . . .	130
show-ConsensusPartition-method . . . . .	131
show-ConsensusPartitionList-method . . . . .	132
show-dispatch . . . . .	132
show-DownSamplingConsensusPartition-method . . . . .	133
show-HierarchicalPartition-method . . . . .	134
split_node-HierarchicalPartition-method . . . . .	134
suggest_best_k-ConsensusPartition-method . . . . .	135
suggest_best_k-ConsensusPartitionList-method . . . . .	137
suggest_best_k-dispatch . . . . .	138
suggest_best_k-HierarchicalPartition-method . . . . .	138
test_between_factors . . . . .	139
test_to_known_factors-ConsensusPartition-method . . . . .	140
test_to_known_factors-ConsensusPartitionList-method . . . . .	141
test_to_known_factors-dispatch . . . . .	142
test_to_known_factors-DownSamplingConsensusPartition-method . . . . .	143
test_to_known_factors-HierarchicalPartition-method . . . . .	144
top_elements_overlap . . . . .	145
top_rows_heatmap-ConsensusPartition-method . . . . .	146
top_rows_heatmap-ConsensusPartitionList-method . . . . .	147
top_rows_heatmap-dispatch . . . . .	148
top_rows_heatmap-HierarchicalPartition-method . . . . .	148
top_rows_heatmap-matrix-method . . . . .	149
top_rows_overlap-ConsensusPartitionList-method . . . . .	150
top_rows_overlap-dispatch . . . . .	151

top_rows_overlap-HierarchicalPartition-method . . . . .	152
top_rows_overlap-matrix-method . . . . .	153

## Index 155

---

[.ConsensusPartitionList

*Subset a ConsensusPartitionList object*

---

### Description

Subset a ConsensusPartitionList object

### Usage

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

### Arguments

x	A <a href="#">ConsensusPartitionList-class</a> object.
i	Index for top-value methods, character or numeric.
j	Index for partitioning methods, character or numeric.
drop	Whether drop class

### Details

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

### Value

A [ConsensusPartitionList-class](#) object or a [ConsensusPartition-class](#) object.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
data(golub_colo)
golub_colo[c("SD", "MAD"), c("hclust", "kmeans")]
golub_colo["SD", "kmeans"] # a ConsensusPartition object
golub_colo["SD:kmeans"] # a ConsensusPartition object
golub_colo[["SD:kmeans"]] # a ConsensusPartition object
golub_colo["SD", "kmeans", drop = FALSE] # still a ConsensusPartitionList object
golub_colo["SD:kmeans", drop = FALSE] # still a ConsensusPartitionList object
golub_colo["SD", ]
golub_colo[, "hclust"]
golub_colo[1:2, 1:2]
```

---

[.HierarchicalPartition

*Subset the HierarchicalPartition object*

---

### Description

Subset the HierarchicalPartition object

### Usage

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

### Arguments

x                   A [HierarchicalPartition-class](#) object.  
i                   Index. The value should be numeric or a node ID.

### Details

On each node, there is a [ConsensusPartition-class](#) object.

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

### Value

A [ConsensusPartition-class](#) object.

### Examples

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

---

[.ConsensusPartitionList

*Subset a ConsensusPartitionList object*

---

### Description

Subset a ConsensusPartitionList object

### Usage

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

**Arguments**

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

**Value**

A [ConsensusPartition-class](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

[[.HierarchicalPartition

*Subset the HierarchicalPartition object*

---

**Description**

Subset the HierarchicalPartition object

**Usage**

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

**Arguments**

- x A [HierarchicalPartition-class](#) object
- i Index. The value should be numeric or a node ID.

**Details**

On each node, there is a [ConsensusPartition-class](#) object.

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

**Value**

A [ConsensusPartition-class](#) object.

**Examples**

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

---

adjust_matrix	<i>Remove rows with low variance and impute missing values</i>
---------------	--

---

**Description**

Remove rows with low variance and impute missing values

**Usage**

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

**Arguments**

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

**Details**

The function uses [impute.knn](#) to impute missing values, then uses [adjust\\_outlier](#) to adjust outliers and removes rows with low standard deviations.

**Value**

A numeric matrix.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

adjust_outlier	<i>Adjust outliers</i>
----------------	------------------------

---

**Description**

Adjust outliers

**Usage**

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

**Arguments**

x	A numeric vector.
q	Percentile to adjust.

**Details**

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

**Value**

A numeric vector with same length as the original one.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

all\_leaves-HierarchicalPartition-method  
*All leaves in the hierarchy*

---

**Description**

All leaves in the hierarchy

**Usage**

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

**Arguments**

object           A [HierarchicalPartition-class](#) object.  
merge\_node       Parameters to merge sub-dendrograms, see [merge\\_node\\_param](#).

**Value**

A vector of node ID.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_colo_rh)  
all_leaves(golub_colo_rh)
```

---

all\_nodes-HierarchicalPartition-method  
*All nodes in the hierarchy*

---

**Description**

All nodes in the hierarchy

**Usage**

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

**Arguments**

object            A [HierarchicalPartition-class](#) object.  
merge\_node       Parameters to merge sub-dendrograms, see [merge\\_node\\_param](#).

**Value**

A vector of node ID.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_colo_rh)
all_nodes(golub_colo_rh)
```

---

all\_partition\_methods *All supported partitioning methods*

---

**Description**

All supported partitioning methods

**Usage**

```
all_partition_methods()
```

**Details**

New partitioning methods can be registered by [register\\_partition\\_methods](#).

**Value**

A vector of supported partitioning methods.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
all_partition_methods()
```

---

all\_top\_value\_methods *All supported top-value methods*

---

**Description**

All supported top-value methods

**Usage**

```
all_top_value_methods()
```

**Details**

New top-value methods can be registered by [register\\_top\\_value\\_methods](#).

**Value**

A vector of supported top-value methods.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
all_top_value_methods()
```

---

aPAC *Adapted PAC scores*

---

**Description**

Adapted PAC scores

**Usage**

```
aPAC(consensus_mat)
```

**Arguments**

consensus\_mat A consensus matrix.

**Details**

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

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

**Value**

A numeric value.

**Examples**

```
data(golub_col1a)
aPAC(get_consensus(golub_col1a[1, 1], k = 2))
aPAC(get_consensus(golub_col1a[1, 1], k = 3))
aPAC(get_consensus(golub_col1a[1, 1], k = 4))
aPAC(get_consensus(golub_col1a[1, 1], k = 5))
aPAC(get_consensus(golub_col1a[1, 1], k = 6))
```

---

 ATC

*Ability to correlate to other rows*


---

**Description**

Ability to correlate to other rows

**Usage**

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

**Arguments**

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

## Details

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

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

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

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

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

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

## Value

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

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## See Also

[https://jokergoo.github.io/cola\\_supplementary/suppl\\_1\\_ATC/suppl\\_1\\_ATC.html](https://jokergoo.github.io/cola_supplementary/suppl_1_ATC/suppl_1_ATC.html)

## Examples

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

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

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

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

---

 ATC\_approx
 

---

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


---

### Description

Ability to correlate to other rows - an approximated method

### Usage

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

### Arguments

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

### Details

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

With small numbers of rows of the matrix, [ATC](#) should be used which calculates the "exact" ATC value, but the value of [ATC](#) and [ATC\\_approx](#) should be very similar.

### Examples

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

---

cola	<i>A bottle of cola</i>
------	-------------------------

---

**Description**

A bottle of cola

**Usage**

```
cola()
```

**Details**

Simply serve you a bottle of cola.

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

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

cola_opt	<i>Global parameters</i>
----------	--------------------------

---

**Description**

Global parameters

**Usage**

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

**Arguments**

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

**Details**

There are following global parameters:

`group_diff` Used in [get\\_signatures, ConsensusPartition-method](#) to globally control the minimal difference between subgroups.

`fdr_cutoff` Used in [get\\_signatures, ConsensusPartition-method](#) to globally control the cut-off of FDR for the differential signature tests.

`color_set_2` Colors for the predicted subgroups.

`help` Whether to print help messages.

`message` Whether to print messages.

**Examples**

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

---

cola\_report-ConsensusPartition-method

*Make HTML report from the ConsensusPartition object*

---

**Description**

Make HTML report from the ConsensusPartition object

**Usage**

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

**Arguments**

<code>object</code>	A <a href="#">ConsensusPartition-class</a> object.
<code>output_dir</code>	The output directory where the report is saved.
<code>title</code>	Title of the report.
<code>env</code>	Where the objects in the report are found, internally used.

**Details**

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

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[cola\\_report](#), [ConsensusPartitionList-method](#)

**Examples**

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

---

cola\_report-ConsensusPartitionList-method

*Make HTML report from the ConsensusPartitionList object*

---

**Description**

Make HTML report from the ConsensusPartitionList object

**Usage**

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

**Arguments**

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

**Details**

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

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

Examples of reports can be found at [https://jokergoo.github.io/cola\\_collection/](https://jokergoo.github.io/cola_collection/).

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

cola\_report-dispatch *Method dispatch page for cola\_report*

---

**Description**

Method dispatch page for cola\_report.

**Dispatch**

cola\_report can be dispatched on following classes:

- [cola\\_report, HierarchicalPartition-method, HierarchicalPartition-class](#) class method
- [cola\\_report, ConsensusPartition-method, ConsensusPartition-class](#) class method
- [cola\\_report, ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method

**Examples**

```
# no example  
NULL
```

---

cola\_report-HierarchicalPartition-method

*Make HTML report from the HierarchicalPartition object*

---

## Description

Make HTML report from the HierarchicalPartition object

## Usage

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

## Arguments

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

## Details

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

## Value

No value is returned.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

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

---

`cola_rl`*Example ConsensusPartitionList object*

---

**Description**

Example ConsensusPartitionList object

**Usage**`data(cola_rl)`**Details**Following code was used to generate `cola_rl`:

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

**Author(s)**

Zuguang Gu &lt;z.gu@dkfz.de&gt;

**Examples**

```
data(cola_rl)
cola_rl
```

---

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

---

**Description**

Collect subgroups from ConsensusPartition object

**Usage**

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

**Arguments**

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

**Details**

The percent membership matrix and the subgroup labels for each k are plotted in the heatmaps. Same row in all heatmaps corresponds to the same column in the original matrix.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_col1)  
collect_classes(golub_col1["ATC", "skmeans"])
```

---

collect\_classes-ConsensusPartitionList-method

*Collect classes from ConsensusPartitionList object*

---

**Description**

Collect classes from ConsensusPartitionList object

**Usage**

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

**Arguments**

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

**Details**

There are following panels in the plot:

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

The row clustering is applied on the dissimilarity matrix calculated by [cl\\_dissimilarity](#) with the comembership method.

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

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_col1)
collect_classes(golub_col1, k = 3)
```

---

collect\_classes-dispatch

*Method dispatch page for collect\_classes*

---

**Description**

Method dispatch page for collect\_classes.

**Dispatch**

collect\_classes can be dispatched on following classes:

- [collect\\_classes, HierarchicalPartition-method, HierarchicalPartition-class](#) class method
- [collect\\_classes, ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method
- [collect\\_classes, ConsensusPartition-method, ConsensusPartition-class](#) class method

**Examples**

```
# no example
NULL
```

---

collect\_classes-HierarchicalPartition-method

*Collect classes from HierarchicalPartition object*

---

**Description**

Collect classes from HierarchicalPartition object

**Usage**

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

**Arguments**

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

**Details**

The function plots the hierarchy of the classes.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

collect\_plots-ConsensusPartition-method

*Collect plots from ConsensusPartition object*

---

**Description**

Collect plots from ConsensusPartition object

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> 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 available k.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[collect\\_plots](#), [ConsensusPartitionList-method](#) collects plots for the [ConsensusPartitionList-class](#) object.

**Examples**

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

---

collect\_plots-ConsensusPartitionList-method

*Collect plots from ConsensusPartitionList object*

---

**Description**

Collect plots from ConsensusPartitionList object

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartitionList-class</a> object from <a href="#">run_all_consensus_partition_methods</a> .
k	Number of subgroups.
fun	Function used to generate plots. Valid functions are <a href="#">consensus_heatmap</a> , <a href="#">plot_ecdf</a> , <a href="#">membership_heatmap</a> , <a href="#">get_signatures</a> and <a href="#">dimension_reduction</a> .
top_value_method	A vector of top-value methods.

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

**Details**

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

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

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[collect\\_plots, ConsensusPartition-method](#) collects plots for a single [ConsensusPartition-class](#) object.

**Examples**

```
data(golub_colo)
collect_plots(colo_r1, k = 3)
collect_plots(colo_r1, k = 3, fun = membership_heatmap)
collect_plots(colo_r1, k = 3, fun = get_signatures)
```

---

collect\_plots-dispatch

*Method dispatch page for collect\_plots*

---

**Description**

Method dispatch page for collect\_plots.

**Dispatch**

collect\_plots can be dispatched on following classes:

- [collect\\_plots, ConsensusPartition-method, ConsensusPartition-class](#) class method
- [collect\\_plots, ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method

**Examples**

```
# no example
NULL
```

---

collect\_stats-ConsensusPartition-method  
*Draw and compare statistics for a single method*

---

**Description**

Draw and compare statistics for a single method

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> 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 <a href="#">ConsensusPartitionList-class</a> object.
k	Number of subgroups.
layout_nrow	Number of rows in the layout
all_stats	Whether to show all statistics that were calculated. Used internally.
...	Other arguments

### Details

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

### Examples

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

---

collect\_stats-dispatch

*Method dispatch page for collect\_stats*

---

### Description

Method dispatch page for collect\_stats.

### Dispatch

collect\_stats can be dispatched on following classes:

- [collect\\_stats,ConsensusPartitionList-method,ConsensusPartitionList-class](#) class method
- [collect\\_stats,ConsensusPartition-method,ConsensusPartition-class](#) class method

### Examples

```
# no example
NULL
```

---

colnames-ConsensusPartition-method  
*Column names of the matrix*

---

**Description**

Column names of the matrix

**Usage**

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

**Arguments**

x                   A [ConsensusPartition-class](#) object.

**Examples**

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

---

colnames-ConsensusPartitionList-method  
*Column names of the matrix*

---

**Description**

Column names of the matrix

**Usage**

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

**Arguments**

x                   A [ConsensusPartitionList-class](#) object.

**Examples**

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

---

colnames-dispatch      *Method dispatch page for colnames*

---

### Description

Method dispatch page for colnames.

### Dispatch

colnames can be dispatched on following classes:

- [colnames, ConsensusPartition-method, ConsensusPartition-class](#) class method
- [colnames, ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method
- [colnames, DownSamplingConsensusPartition-method, DownSamplingConsensusPartition-class](#) class method
- [colnames, HierarchicalPartition-method, HierarchicalPartition-class](#) class method

### Examples

```
# no example  
NULL
```

---

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

---

### Description

Column names of the matrix

### Usage

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

### Arguments

x                      A [DownSamplingConsensusPartition-class](#) object.

### Examples

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

---

colnames-HierarchicalPartition-method  
*Column names of the matrix*

---

**Description**

Column names of the matrix

**Usage**

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

**Arguments**

x                   A [HierarchicalPartition-class](#) object.

**Examples**

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

---

compare\_partitions-ConsensusPartition-method  
*Compare two partitionings*

---

**Description**

Compare two partitionings

**Usage**

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

**Arguments**

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

**Details**

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

In the report, there are three heatmaps which visualize A) the signature genes specific in the first partition, B) the signature genes both in the two partitionings and C) the signatures genes specific in the second partition. Argument row\_km1, row\_km2 and row\_km3 control how many k-means groups should be applied on the three heatmaps.

**Examples**

```
## Not run:
data(golub_col1)
require(hu6800.db)
x = hu6800ENTREZID
mapped_probes = mappedkeys(x)
id_mapping = unlist(as.list(x[mapped_probes]))
compare_partitions(golub_col1["ATC:skmeans"], golub_col1["SD:kmeans"],
  id_mapping = id_mapping)

## End(Not run)
```

---

compare\_signatures-ConsensusPartition-method

*Compare Signatures from Different k*

---

**Description**

Compare Signatures from Different k

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> object.
k	Number of subgroups. Value should be a vector.
verbose	Whether to print message.
...	Other arguments passed to <a href="#">get_signatures, ConsensusPartition-method</a> .

**Details**

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

**Examples**

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

---

compare\_signatures-dispatch

*Method dispatch page for compare\_signatures*

---

**Description**

Method dispatch page for compare\_signatures.

**Dispatch**

compare\_signatures can be dispatched on following classes:

- [compare\\_signatures, HierarchicalPartition-method, HierarchicalPartition-class](#) class method
- [compare\\_signatures, ConsensusPartition-method, ConsensusPartition-class](#) class method

**Examples**

```
# no example  
NULL
```

---

compare\_signatures-HierarchicalPartition-method  
*Compare Signatures from Different Nodes*

---

### Description

Compare Signatures from Different Nodes

### Usage

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

### Arguments

object	A <a href="#">HierarchicalPartition-class</a> object.
merge_node	Parameters to merge sub-dendrograms, see <a href="#">merge_node_param</a> .
method	Method to visualize.
upset_max_comb_sets	Maximal number of combination sets to show.
verbose	Whether to print message.
...	Other arguments passed to <a href="#">get_signatures, HierarchicalPartition-method</a> .

### Details

It plots an Euler diagram or a UpSet plot showing the overlap of signatures from different nodes. On each node, the number of subgroups is inferred by [suggest\\_best\\_k, ConsensusPartition-method](#).

### Examples

```
data(golub_col4_rh)
compare_signatures(golub_col4_rh)
```

---

concordance *Concordance to the consensus partition*

---

### Description

Concordance to the consensus partition

### Usage

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

**Arguments**

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

**Details**

Note subgroup labels in `membership_each` should already be adjusted to the consensus labels, i.e. by [relabel\\_class](#).

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

**Value**

A numeric value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

config\_ATC

*Adjust parameters for default ATC method*

---

**Description**

Adjust parameters for default ATC method

**Usage**

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

**Arguments**

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

group	A categorical variable.
cores	Number of cores.
...	Other arguments passed to <a href="#">ATC</a> .

### Details

This function changes the default parameters for ATC method. All the arguments in this function all pass to [ATC](#).

### Examples

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

---

consensus\_heatmap-ConsensusPartition-method  
*Heatmap of the consensus matrix*

---

### Description

Heatmap of the consensus matrix

### Usage

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

### Arguments

object	A <a href="#">ConsensusPartition-class</a> object.
k	Number of subgroups.
internal	Used internally.
anno	A data frame of annotations for the original matrix columns. By default it uses the annotations specified in <a href="#">consensus_partition</a> or <a href="#">run_all_consensus_partition_methods</a> .
anno_col	A list of colors (color is defined as a named vector) for the annotations. If anno is a data frame, anno_col should be a named list where names correspond to the column names in anno.
show_row_names	Whether plot row names on the consensus heatmap (which are the column names in the original matrix)
show_column_names	Whether show column names.

row_names_gp	Graphics parameters for row names.
simplify	Internally used.
...	other arguments.

### Details

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

There are following heatmaps from left to right:

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

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

### Value

No value is returned.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### See Also

[membership\\_heatmap, ConsensusPartition-method](#)

### Examples

```
data(golub_col1)
consensus_heatmap(golub_col1["ATC", "skmeans"], k = 3)
```

---

consensus\_partition    *Consensus partition*

---

## Description

Consensus partition

## Usage

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

## Arguments

data	A numeric matrix where subgroups are found by columns.
top_value_method	A single top-value method. Available methods are in <a href="#">all_top_value_methods</a> . Use <a href="#">register_top_value_methods</a> 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, partition will be applied to every values in top_n and consensus partition is summarized from all partitions.
partition_method	A single partitioning method. Available methods are in <a href="#">all_partition_methods</a> . Use <a href="#">register_partition_methods</a> to add a new partition method.
max_k	Maximal number of subgroups to try. The function will try for 2:max_k subgroups
k	Alternatively, you can specify a vector k.
sample_by	Should randomly sample the matrix by rows or by columns?

p_sampling	Proportion of the submatrix which contains the top n rows to sample.
partition_repeat	Number of repeats for the random sampling.
partition_param	Parameters for the partition method which are passed to ... in a registered partitioning method. See <a href="#">register_partition_methods</a> 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 <a href="#">register_partition_methods</a> is used.
verbose	Whether print messages.
mc.cores	Multiple cores to use. This argument will be removed in future versions.
cores	Number of cores, or a cluster object returned by <a href="#">makeCluster</a> .
prefix	Internally used.
.env	An environment, internally used.
help	Whether to print help messages.

### Details

The function performs analysis in following steps:

- calculate scores for rows by top-value method,
- for each top\_n value, take top n rows,
- randomly sample p\_sampling rows from the top\_n-row matrix and perform partitioning for partition\_repeats times,
- collect partitions from all individual partitions and summarize a consensus partition.

### Value

A [ConsensusPartition-class](#) object. Simply type object in the interactive R session to see which functions can be applied on it.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### See Also

[run\\_all\\_consensus\\_partition\\_methods](#) runs consensus partitioning with multiple top-value methods and multiple partitioning methods.

**Examples**

```

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

```

---

consensus\_partition\_by\_down\_sampling

*Consensus partitioning only with a subset of columns*

---

**Description**

Consensus partitioning only with a subset of columns

**Usage**

```

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

```

**Arguments**

data	A numeric matrix where subgroups are found by columns.
top_value_method	A single top-value method. Available methods are in <a href="#">all_top_value_methods</a> . Use <a href="#">register_top_value_methods</a> 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, partition will be applied to every values in top_n and consensus partition is summarized from all partitions.

partition_method	A single partitioning method. Available methods are in <a href="#">all_partition_methods</a> . Use <a href="#">register_partition_methods</a> to add a new partition method.
max_k	Maximal number of subgroups to try. The function will try for 2:max_k subgroups
k	Alternatively, you can specify a vector k.
subset	Number of columns to randomly sample, or a vector of selected indices.
pre_select	Whether to pre-select by k-means.
verbose	Whether to print messages.
prefix	Internally used.
anno	Annotation data frame.
anno_col	Annotation colors.
predict_method	Method for predicting class labels. Possible values are "centroid", "svm" and "randomForest".
dist_method	Method for predict the class for other columns.
.env	An environment, internally used.
.predict	Internally used.
mc.cores	Number of cores. This argument will be removed in future versions.
cores	Number of cores, or a cluster object returned by <a href="#">makeCluster</a> .
...	All pass to <a href="#">consensus_partition</a> .

## Details

The function performs consensus partitioning only with a small subset of columns and the class of other columns are predicted by [predict\\_classes](#), [ConsensusPartition-method](#).

## Examples

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

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

## End(Not run)
```

---

ConsensusPartition-class

*The ConsensusPartition class*

---

## Description

The ConsensusPartition class

## Methods

The `ConsensusPartition-class` has following methods:

- `consensus_partition`: constructor method, run consensus partitioning with a specified top-value method and a partitioning method.
- `select_partition_number, ConsensusPartition-method`: make a list of plots for selecting optimized number of subgroups.
- `consensus_heatmap, ConsensusPartition-method`: make heatmap of the consensus matrix.
- `membership_heatmap, ConsensusPartition-method`: make heatmap of the membership for individual partitions.
- `get_signatures, ConsensusPartition-method`: get the signature rows and make heatmap.
- `dimension_reduction, ConsensusPartition-method`: make dimension reduction plots.
- `collect_plots, ConsensusPartition-method`: make heatmaps for consensus matrix and membership matrix with different number of subgroups.
- `collect_classes, ConsensusPartition-method`: make heatmap with subgroups with different numbers.
- `get_param, ConsensusPartition-method`: get parameters for the consensus clustering.
- `get_matrix, ConsensusPartition-method`: get the original matrix.
- `get_consensus, ConsensusPartition-method`: get the consensus matrix.
- `get_membership, ConsensusPartition-method`: get the membership of partitions generated from random samplings.
- `get_stats, ConsensusPartition-method`: get statistics for the consensus partitioning.
- `get_classes, ConsensusPartition-method`: get the consensus subgroup labels and other columns.
- `suggest_best_k, ConsensusPartition-method`: guess the best number of subgroups.
- `test_to_known_factors, ConsensusPartition-method`: test correlation between predicted subgroups and known factors, if available.
- `cola_report, ConsensusPartition-method`: generate a HTML report for the whole analysis.
- `functional_enrichment, ConsensusPartition-method`: perform functional enrichment analysis on significant genes if rows in the matrix can be corresponded to genes.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

ConsensusPartitionList-class

*The ConsensusPartitionList class*

---

**Description**

The ConsensusPartitionList class

**Details**

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

**Methods**

The `ConsensusPartitionList-class` provides following methods:

`run_all_consensus_partition_methods`: constructor method.

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

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

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

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

`get_stats, ConsensusPartitionList-method`: get statistics for the partition for a specified k.

`get_membership, ConsensusPartitionList-method`: get consensus membership matrix summarized from all methods.

`suggest_best_k, ConsensusPartitionList-method`: guess the best number of subgroups for all methods.

`collect_plots, ConsensusPartitionList-method`: collect plots from all combinations of top-value methods and partitioning methods with choosing a plotting function.

`collect_classes, ConsensusPartitionList-method`: make a plot which contains predicted subgroups from all combinations of top-value methods and partitioning methods.

`test_to_known_factors, ConsensusPartitionList-method`: test correlation between predicted subgroups and known annotations, if provided.

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

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

The [ConsensusPartition-class](#).

**Examples**

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

---

correspond\_between\_rankings

*Correspond between a list of rankings*

---

**Description**

Correspond between a list of rankings

**Usage**

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

**Arguments**

lt	A list of scores under different metrics.
top_n	Top n elements to show the correspondance.
col	A vector of colors for lt.
...	Pass to <a href="#">correspond_between_two_rankings</a> .

**Details**

It makes plots for every pairwise comparison in lt.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

## Examples

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

---

correspond\_between\_two\_rankings  
*Correspond two rankings*

---

## Description

Correspond two rankings

## Usage

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

## Arguments

x1	A vector of scores calculated by one metric.
x2	A vector of scores calculated by another metric.
name1	Name of the first metric.
name2	Name of the second metric.
col1	Color for the first metric.
col2	Color for the second metric.
top_n	Top n elements to show the correspondance.
transparency	Transparency of the connecting lines.
pt_size	Size of the points, must be a <code>unit</code> 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  $x_1$  and  $x_2$ , the  $i^{\text{th}}$  element in both vectors corresponds to the same object (e.g. same row if they are calculated from a matrix) but with different scores under different metrics.

$x_1$  and  $x_2$  are sorted in the left panel and right panel respectively. The top  $n$  elements under corresponding metric are highlighted by vertical colored lines in both panels. The left and right panels also shown as barplots of the scores in the two metrics. Between the left and right panels, there are lines connecting the same element (e.g.  $i^{\text{th}}$  element in  $x_1$  and  $x_2$ ) in the two ordered vectors so that you can see how a same element has two different ranks in the two metrics.

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

## Value

No value is returned.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## See Also

[correspond\\_between\\_rankings](#) draws for more than 2 sets of rankings.

## Examples

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

---

david\_enrichment

*Perform DAVID enrichment analysis*

---

## Description

Perform DAVID enrichment analysis

## Usage

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

## Arguments

genes	A vector of gene identifiers.
email	The email that user registered on DAVID web service ( <a href="https://david.ncifcrf.gov/content.jsp?file=WS.html">https://david.ncifcrf.gov/content.jsp?file=WS.html</a> ).
catalog	A vector of function catalogs. Valid values should be in <code>cola:::DAVID_ALL_CATALOGS</code> .
idtype	ID types for the input gene list. Valid values should be in <code>cola:::DAVID_ALL_ID_TYPES</code> .
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](mailto:z.gu@dkfz.de)>

## See Also

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

## Examples

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

dim.ConsensusPartition  
*Dimension of the matrix*

---

**Description**

Dimension of the matrix

**Usage**

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

**Arguments**

x                    A [ConsensusPartition-class](#) object.

**Examples**

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

---

dim.ConsensusPartitionList  
*Dimension of the matrix*

---

**Description**

Dimension of the matrix

**Usage**

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

**Arguments**

x                    A [ConsensusPartitionList-class](#) object.

**Examples**

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

---

```
dim.DownSamplingConsensusPartition  
    Dimension of the matrix
```

---

### **Description**

Dimension of the matrix

### **Usage**

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

### **Arguments**

x                    A [DownSamplingConsensusPartition-class](#) object.

### **Examples**

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

---

```
dim.HierarchicalPartition  
    Dimension of the matrix
```

---

### **Description**

Dimension of the matrix

### **Usage**

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

### **Arguments**

x                    A [HierarchicalPartition-class](#) object.

### **Examples**

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

---

 dimension\_reduction-ConsensusPartition-method

*Visualize column after dimension reduction*


---

### Description

Visualize samples (the matrix columns) after dimension reduction

### Usage

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

### Arguments

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

### Value

Locations of the points.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

```
dimension_reduction-dispatch
```

*Method dispatch page for dimension\_reduction*

---

**Description**

Method dispatch page for dimension\_reduction.

**Dispatch**

dimension\_reduction can be dispatched on following classes:

- [dimension\\_reduction, ConsensusPartition-method, ConsensusPartition-class](#) class method
- [dimension\\_reduction, DownSamplingConsensusPartition-method, DownSamplingConsensusPartition-class](#) class method
- [dimension\\_reduction, HierarchicalPartition-method, HierarchicalPartition-class](#) class method
- [dimension\\_reduction, matrix-method, matrix-class](#) class method

**Examples**

```
# no example
NULL
```

---

```
dimension_reduction-DownSamplingConsensusPartition-method
```

*Visualize column after dimension reduction*

---

**Description**

Visualize samples (the matrix columns) after dimension reduction

**Usage**

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

**Arguments**

object	A <a href="#">DownSamplingConsensusPartition-class</a> object.
k	Number of subgroups.
top_n	Top n rows to use. By default it uses all rows in the original matrix.
method	Which method to reduce the dimension of the data. MDS uses <a href="#">cmdscale</a> , PCA uses <a href="#">prcomp</a> . t-SNE uses <a href="#">Rtsne</a> . UMAP uses <a href="#">umap</a> .
color_by	If annotation table is set, an annotation name can be set here.
control	A list of parameters for <a href="#">Rtsne</a> or <a href="#">umap</a> .
internal	Internally used.
nr	If number of matrix rows is larger than this value, random nr rows are used.
p_cutoff	Cutoff of p-value of class label prediction. Data points with values higher than it will be mapped with cross symbols.
remove	Whether to remove columns which have high p-values than the cutoff.
scale_rows	Whether to perform scaling on matrix rows.
verbose	Whether print messages.
...	Other arguments.

**Details**

This function is basically very similar as [dimension\\_reduction,ConsensusPartition-method](#).

**Value**

No value is returned.

**Examples**

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

---

dimension\_reduction-HierarchicalPartition-method

*Visualize columns after dimension reduction*

---

**Description**

Visualize columns after dimension reduction

**Usage**

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

**Arguments**

object	A <a href="#">HierarchicalPartition-class</a> object.
merge_node	Parameters to merge sub-dendrograms, see <a href="#">merge_node_param</a> .
top_n	Top n rows to use. By default it uses all rows in the original matrix.
top_value_method	Which top-value method to use.
parent_node	Parent node. If it is set, the function call is identical to <code>dimension_reduction(object[parent_node])</code>
method	Which method to reduce the dimension of the data. MDS uses <a href="#">cmdscale</a> , PCA uses <a href="#">prcomp</a> . t-SNE uses <a href="#">Rtsne</a> . UMAP uses <a href="#">umap</a> .
color_by	If annotation table is set, an annotation name can be set here.
scale_rows	Whether to perform scaling on matrix rows.
verbose	Whether print messages.
...	Other arguments passed to <a href="#">dimension_reduction,ConsensusPartition-method</a> .

**Details**

The class IDs are extract at depth.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <[z.gu@dkfz.de](mailto:z.gu@dkfz.de)>

**Examples**

```
data(golub_colo_rh)
dimension_reduction(golub_colo_rh)
```

---

dimension\_reduction-matrix-method

*Visualize columns after dimension reduction*

---

**Description**

Visualize columns after dimension reduction

**Usage**

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

**Arguments**

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

**Value**

Locations of the points.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

 DownSamplingConsensusPartition-class

*The DownSamplingConsensusPartition class*


---

### Description

The DownSamplingConsensusPartition class

### Details

The DownSamplingConsensusPartition performs consensus partitioning only with a small subset of columns and the class of other columns are predicted by [predict\\_classes, ConsensusPartition-method](#).

The DownSamplingConsensusPartition-class is a child class of [ConsensusPartition-class](#). It inherits all methods of [ConsensusPartition-class](#).

### See Also

The constructor function [consensus\\_partition\\_by\\_down\\_sampling](#).

### Examples

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

---

 FCC

*Flatness of the CDF curve*


---

### Description

Flatness of the CDF curve

### Usage

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

### Arguments

consensus\_mat    A consensus matrix.  
diff              Difference of  $F(b) - F(a)$ .

### Details

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

**Value**

A numeric value.

**Examples**

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

---

find\_best\_km

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

---

**Description**

Find a best k for the k-means clustering

**Usage**

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

**Arguments**

mat                    A matrix where k-means clustering is executed by rows.  
max\_km                Maximal k to try.

**Details**

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

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

**Examples**

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

---

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

---

**Description**

Perform functional enrichment on signature genes

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

A data frame.

**See Also**

[http://bioconductor.org/packages/devel/bioc/vignettes/cola/inst/doc/functional\\_enrichment.html](http://bioconductor.org/packages/devel/bioc/vignettes/cola/inst/doc/functional_enrichment.html)

**Examples**

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

---

```
functional_enrichment-ConsensusPartition-method
Perform functional enrichment on signature genes
```

---

**Description**

Perform functional enrichment on signature genes

**Usage**

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

**Arguments**

object	a <a href="#">ConsensusPartition-class</a> object from <a href="#">run_all_consensus_partition_methods</a> .
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 corresponding Entrez IDs. The value can also be a function that converts gene IDs.
org_db	Annotation database.
ontology	See corresponding argument in <a href="#">functional_enrichment, ANY-method</a> .
min_set_size	The minimal size of the gene sets.
max_set_size	The maximal size of the gene sets.
verbose	Whether to print messages.
prefix	Used internally.
...	Pass to <a href="#">functional_enrichment, ANY-method</a> .

**Details**

For how to control the parameters of functional enrichment, see help page of [functional\\_enrichment, ANY-method](#).

**Value**

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

**See Also**

[http://bioconductor.org/packages/devel/bioc/vignettes/cola/inst/doc/functional\\_enrichment.html](http://bioconductor.org/packages/devel/bioc/vignettes/cola/inst/doc/functional_enrichment.html)

**Examples**

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

---

functional\_enrichment-ConsensusPartitionList-method  
*Perform functional enrichment on signature genes*

---

**Description**

Perform functional enrichment on signature genes

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartitionList-class</a> object from <a href="#">run_all_consensus_partition_methods</a> .
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 corresponding Entrez IDs. The value can also be a function that converts gene IDs.
org_db	Annotation database.
ontology	See corresponding argument in <a href="#">functional_enrichment, ANY-method</a> .
min_set_size	The minimal size of the gene sets.
max_set_size	The maximal size of the gene sets.
...	Pass to <a href="#">functional_enrichment, ANY-method</a> .

**Details**

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

It calls [functional\\_enrichment,ConsensusPartition-method](#) on the consensus partitioning results for each method.

For how to control the parameters of functional enrichment, see help page of [functional\\_enrichment,ANY-method](#).

**Value**

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

**See Also**

[http://bioconductor.org/packages/devel/bioc/vignettes/cola/inst/doc/functional\\_enrichment.html](http://bioconductor.org/packages/devel/bioc/vignettes/cola/inst/doc/functional_enrichment.html)

**Examples**

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

---

functional\_enrichment-dispatch

*Method dispatch page for functional\_enrichment*

---

**Description**

Method dispatch page for functional\_enrichment.

**Dispatch**

functional\_enrichment can be dispatched on following classes:

- [functional\\_enrichment,HierarchicalPartition-method,HierarchicalPartition-class](#) class method
- [functional\\_enrichment,ANY-method,ANY-class](#) class method
- [functional\\_enrichment,ConsensusPartition-method,ConsensusPartition-class](#) class method
- [functional\\_enrichment,ConsensusPartitionList-method,ConsensusPartitionList-class](#) class method

**Examples**

```
# no example
NULL
```

---

```
functional_enrichment-HierarchicalPartition-method
  Perform functional enrichment on signature genes
```

---

**Description**

Perform functional enrichment on signature genes

**Usage**

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

**Arguments**

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

**Details**

For how to control the parameters of functional enrichment, see help page of [functional\\_enrichment, ANY-method](#).

**Value**

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

**Examples**

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

---

get\_anno-ConsensusPartition-method  
*Get annotations*

---

**Description**

Get annotations

**Usage**

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

**Arguments**

object            A [ConsensusPartition-class](#) object.

**Value**

A data frame if anno was specified in [run\\_all\\_consensus\\_partition\\_methods](#) or [consensus\\_partition](#), or else NULL.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_anno-ConsensusPartitionList-method  
*Get annotations*

---

### Description

Get annotations

### Usage

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

### Arguments

object            A [ConsensusPartitionList-class](#) object.

### Value

A data frame if anno was specified in [run\\_all\\_consensus\\_partition\\_methods](#), or else NULL.

### Author(s)

Zuguang Gu <[z.gu@dkfz.de](mailto:z.gu@dkfz.de)>

### Examples

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

---

get\_anno-dispatch        *Method dispatch page for get\_anno*

---

### Description

Method dispatch page for get\_anno.

### Dispatch

get\_anno can be dispatched on following classes:

- [get\\_anno,HierarchicalPartition-method](#), [HierarchicalPartition-class](#) class method
- [get\\_anno,ConsensusPartition-method](#), [ConsensusPartition-class](#) class method
- [get\\_anno,ConsensusPartitionList-method](#), [ConsensusPartitionList-class](#) class method
- [get\\_anno,DownSamplingConsensusPartition-method](#), [DownSamplingConsensusPartition-class](#) class method

**Examples**

```
# no example  
NULL
```

---

```
get_anno-DownSamplingConsensusPartition-method  
Get annotations
```

---

**Description**

Get annotations

**Usage**

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

**Arguments**

object	A <a href="#">DownSamplingConsensusPartition-class</a> object.
reduce	Used internally.

**Value**

A data frame if anno was specified in [consensus\\_partition\\_by\\_down\\_sampling](#), or else NULL.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_colo_ds)  
get_anno(golub_colo_ds)
```

---

get\_anno-HierarchicalPartition-method  
*Get annotations*

---

**Description**

Get annotations

**Usage**

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

**Arguments**

object            A [HierarchicalPartition-class](#) object.

**Value**

A data frame if anno was specified in [hierarchical\\_partition](#), or NULL.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_anno\_col-ConsensusPartition-method  
*Get annotation colors*

---

**Description**

Get annotation colors

**Usage**

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

**Arguments**

object            A [ConsensusPartition-class](#) object.

**Value**

A list of color vectors or else NULL.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

*get\_anno\_col-ConsensusPartitionList-method*  
*Get annotation colors*

---

**Description**

Get annotation colors

**Usage**

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

**Arguments**

object            A [ConsensusPartitionList-class](#) object.

**Value**

A list of color vectors or else NULL.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_anno\_col-dispatch *Method dispatch page for get\_anno\_col*

---

**Description**

Method dispatch page for get\_anno\_col.

**Dispatch**

get\_anno\_col can be dispatched on following classes:

- [get\\_anno\\_col, HierarchicalPartition-method, HierarchicalPartition-class](#) class method
- [get\\_anno\\_col, ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method
- [get\\_anno\\_col, ConsensusPartition-method, ConsensusPartition-class](#) class method

**Examples**

```
# no example  
NULL
```

---

get\_anno\_col-HierarchicalPartition-method  
*Get annotation colors*

---

**Description**

Get annotation colors

**Usage**

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

**Arguments**

object            A [HierarchicalPartition-class](#) object.

**Value**

A list of color vectors or NULL.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

```
get_children_nodes-HierarchicalPartition-method
    Test whether a node is a leaf node
```

---

**Description**

Test whether a node is a leaf node

**Usage**

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

**Arguments**

object	A <a href="#">HierarchicalPartition-class</a> object.
node	A vector of node IDs.
merge_node	Parameters to merge sub-dendrograms, see <a href="#">merge_node_param</a> .

**Value**

A vector of children nodes.

**Examples**

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

---

```
get_classes-ConsensusPartition-method
    Get subgroup labels
```

---

**Description**

Get subgroup labels

**Usage**

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

**Arguments**

object            A [ConsensusPartition-class](#) object.  
k                 Number of subgroups.

**Value**

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

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_classes-ConsensusPartitionList-method  
*Get subgroup labels*

---

**Description**

Get subgroup labels

**Usage**

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

**Arguments**

object            A [ConsensusPartitionList-class](#) object.  
k                 Number of subgroups.

**Details**

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

**Value**

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_classes-dispatch *Method dispatch page for get\_classes*

---

**Description**

Method dispatch page for get\_classes.

**Dispatch**

get\_classes can be dispatched on following classes:

- [get\\_classes, HierarchicalPartition-method, HierarchicalPartition-class](#) class method
- [get\\_classes, ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method
- [get\\_classes, ConsensusPartition-method, ConsensusPartition-class](#) class method
- [get\\_classes, DownSamplingConsensusPartition-method, DownSamplingConsensusPartition-class](#) class method

**Examples**

```
# no example
NULL
```

---

get\_classes-DownSamplingConsensusPartition-method  
*Get subgroup labels*

---

**Description**

Get subgroup labels

**Usage**

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

**Arguments**

object	A <a href="#">DownSamplingConsensusPartition-class</a> object.
k	Number of subgroups.
p_cutoff	Cutoff of p-values of class label prediction. It is only used when k is a vector.
reduce	Used internally.

**Value**

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

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

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

**Examples**

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

---

get\_classes-HierarchicalPartition-method

*Get class IDs from the HierarchicalPartition object*

---

**Description**

Get class IDs from the HierarchicalPartition object

**Usage**

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

**Arguments**

object	A <a href="#">HierarchicalPartition-class</a> object.
merge_node	Parameters to merge sub-dendrograms, see <a href="#">merge_node_param</a> .

**Value**

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_consensus-ConsensusPartition-method  
*Get consensus matrix*

---

**Description**

Get consensus matrix

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> object.
k	Number of subgroups.

**Details**

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

**Value**

A consensus matrix corresponding to the current  $k$ .

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_matrix-ConsensusPartition-method  
*Get the original matrix*

---

**Description**

Get the original matrix

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> object.
full	Whether to extract the complete original matrix.
include_all_rows	Internally used.

**Value**

A numeric matrix.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_col4)  
obj = golub_col4["ATC", "skmeans"]  
get_matrix(obj)
```

---

get\_matrix-ConsensusPartitionList-method  
*Get the original matrix*

---

**Description**

Get the original matrix

**Usage**

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

**Arguments**

object            A [ConsensusPartitionList-class](#) object.

**Value**

A numeric matrix.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_matrix-dispatch    *Method dispatch page for get\_matrix*

---

**Description**

Method dispatch page for get\_matrix.

**Dispatch**

get\_matrix can be dispatched on following classes:

- [get\\_matrix,ConsensusPartition-method,ConsensusPartition-class](#) class method
- [get\\_matrix,ConsensusPartitionList-method,ConsensusPartitionList-class](#) class method
- [get\\_matrix,DownSamplingConsensusPartition-method,DownSamplingConsensusPartition-class](#) class method
- [get\\_matrix,HierarchicalPartition-method,HierarchicalPartition-class](#) class method

**Examples**

```
# no example
NULL
```

---

*get\_matrix-DownSamplingConsensusPartition-method*  
*Get the original matrix*

---

**Description**

Get the original matrix

**Usage**

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

**Arguments**

object           A [DownSamplingConsensusPartition-class](#) object.  
reduce           Whether to return the reduced matrix where columns are randomly sampled.

**Value**

A numeric matrix

**Examples**

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

---

*get\_matrix-HierarchicalPartition-method*  
*Get the original matrix*

---

**Description**

Get the original matrix

**Usage**

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

**Arguments**

object           A [HierarchicalPartition-class](#) object.

**Value**

A numeric matrix.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

```
get_membership-ConsensusPartition-method
Get membership matrix
```

---

**Description**

Get membership matrix

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> object.
k	Number of subgroups.
each	Whether to return the percentage membership matrix which is summarized from all partitions or the individual membership in every single partition run.

**Details**

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

The percent membership matrix is calculated by [cl\\_consensus](#).

**Value**

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[get\\_membership, ConsensusPartitionList-method](#) summarizes membership from partitions from all combinations of top-value methods and partitioning methods.

**Examples**

```
data(golub_col1a)
obj = golub_col1a["ATC", "skmeans"]
get_membership(obj, k = 2)
get_membership(obj, k = 2, each = TRUE)
```

---

get\_membership-ConsensusPartitionList-method  
*Get membership matrix*

---

**Description**

Get membership matrix

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartitionList-class</a> object.
k	Number of subgroups.

**Details**

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

**Value**

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[get\\_membership, ConsensusPartition-method](#) returns membership matrix for a single top-value method and partitioning method.

**Examples**

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

---

[get\\_membership-dispatch](#)

*Method dispatch page for get\_membership*

---

**Description**

Method dispatch page for get\_membership.

**Dispatch**

get\_membership can be dispatched on following classes:

- [get\\_membership, ConsensusPartition-method, ConsensusPartition-class](#) class method
- [get\\_membership, ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method

**Examples**

```
# no example
NULL
```

---

[get\\_param-ConsensusPartition-method](#)

*Get parameters*

---

**Description**

Get parameters

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> object.
k	Number of subgroups.
unique	Whether to apply <a href="#">unique</a> to rows of the returned data frame.

**Details**

It is mainly used internally.

**Value**

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_signatures-ConsensusPartition-method  
*Get signature rows*

---

**Description**

Get signature rows

**Usage**

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

```

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

```

### Arguments

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

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

## Details

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

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

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

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

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

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

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

**Value**

A data frame with more than two columns:

**which\_row**: row index corresponding to the original matrix.

**fdr**: the FDR.

**km**: the k-means groups if row\_km is set.

**other\_columns**: the mean value (depending rows are scaled or not) in each subgroup.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_signatures-dispatch

*Method dispatch page for get\_signatures*

---

**Description**

Method dispatch page for get\_signatures.

**Dispatch**

get\_signatures can be dispatched on following classes:

- [get\\_signatures,ConsensusPartition-method,ConsensusPartition-class](#) class method
- [get\\_signatures,DownSamplingConsensusPartition-method,DownSamplingConsensusPartition-class](#) class method
- [get\\_signatures,HierarchicalPartition-method,HierarchicalPartition-class](#) class method

**Examples**

```
# no example
NULL
```

---

get\_signatures-DownSamplingConsensusPartition-method  
*Get signature rows*

---

### Description

Get signature rows

### Usage

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

### Arguments

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

### Details

This function is very similar as [get\\_signatures,ConsensusPartition-method](#).

### Examples

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

---

get\_signatures-HierarchicalPartition-method  
*Get signatures rows*

---

### Description

Get signatures rows

**Usage**

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

**Arguments**

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

**Details**

The function calls [get\\_signatures,ConsensusPartition-method](#) to find signatures at each node of the partition hierarchy.

**Value**

A data frame with more than two columns:

**which\_row:** row index corresponding to the original matrix.

**km:** the k-means groups if row\_km is set.

**other\_columns:** the mean value (depending rows are scaled or not) in each subgroup.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_stats-ConsensusPartition-method  
*Get statistics*

---

**Description**

Get statistics

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> object.
k	Number of subgroups. The value can be a vector.
all_stats	Whether to show all statistics that were calculated. Used internally.

**Details**

The statistics are:

**1-PAC** 1 - proportion of ambiguous clustering, calculated by [PAC](#).

**mean\_silhouette** The mean silhouette score. See [https://en.wikipedia.org/wiki/Silhouette\\_\(clustering\)](https://en.wikipedia.org/wiki/Silhouette_(clustering)).

**concordance** The mean probability that each partition fits the consensus partition, calculated by [concordance](#).

**area\_increased** The increased area under eCDF (the empirical cumulative distribution function) curve to the previous k.

**Rand** This is the percent of pairs of samples that are both in a same cluster or both are not in a same cluster in the partition of k and k-1. See [https://en.wikipedia.org/wiki/Rand\\_index](https://en.wikipedia.org/wiki/Rand_index).

**Jaccard** The ratio of pairs of samples are both in a same cluster in the partition of k and k-1 and the pairs of samples are both in a same cluster in the partition k or k-1.

### Value

A matrix of partition statistics.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

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

---

get\_stats-ConsensusPartitionList-method  
*Get statistics*

---

### Description

Get statistics

### Usage

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

### Arguments

object	A <a href="#">ConsensusPartitionList-class</a> object.
k	Number of subgroups. The value can only be a single value.
all_stats	Whether to show all statistics that were calculated. Used internally.

### Value

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

get\_stats-dispatch      *Method dispatch page for get\_stats*

---

**Description**

Method dispatch page for get\_stats.

**Dispatch**

get\_stats can be dispatched on following classes:

- [get\\_stats,ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method
- [get\\_stats,ConsensusPartition-method, ConsensusPartition-class](#) class method

**Examples**

```
# no example
NULL
```

---

golub\_colo      *Example ConsensusPartitionList object from Golub dataset*

---

**Description**

Example ConsensusPartitionList object from Golub dataset

**Usage**

```
data(golub_colo)
```

## Details

Following code was used to generate golub\_colo:

```
library(colo)

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

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

m = adjust_matrix(m)

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

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

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## See Also

[https://jokergoo.github.io/colo\\_examples/Golub\\_leukemia/](https://jokergoo.github.io/colo_examples/Golub_leukemia/)

## Examples

```
data(golub_colo)
golub_colo
```

---

golub_colo_ds	<i>Example DownSamplingConsensusPartition object from Golub dataset</i>
---------------	---

---

**Description**

Example DownSamplingConsensusPartition object from Golub dataset

**Usage**

```
data(golub_colo_ds)
```

**Details**

Following code was used to generate golub\_colo\_ds:

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_colo_ds)
golub_colo_ds
```

---

golub_colo_rh	<i>Example HierarchicalPartition object from Golub dataset</i>
---------------	--

---

**Description**

Example HierarchicalPartition object from Golub dataset

**Usage**

```
data(golub_colo_rh)
```

**Details**

Following code was used to generate golub\_colo\_rh:

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_colo_rh)
golub_colo_rh
```

---

hierarchical\_partition

*Hierarchical partition*

---

**Description**

Hierarchical partition

**Usage**

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

**Arguments**

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

**Details**

The function looks for subgroups in a hierarchical way.

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

**Value**

A [HierarchicalPartition-class](#) object. Simply type object in the interactive R session to see which functions can be applied on it.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

## End(Not run)
```

---

HierarchicalPartition-class

*The HierarchicalPartition class*

---

**Description**

The HierarchicalPartition class

**Methods**

The [HierarchicalPartition-class](#) has following methods:

[hierarchical\\_partition](#): constructor method.

[collect\\_classes](#), [HierarchicalPartition-method](#): plot the hierarchy of subgroups predicted.

[get\\_classes](#), [HierarchicalPartition-method](#): get the class IDs of subgroups.

[suggest\\_best\\_k,HierarchicalPartition-method](#): guess the best number of partitions for each node.

[get\\_matrix,HierarchicalPartition-method](#): get the original matrix.

[get\\_signatures,HierarchicalPartition-method](#): get the signatures for each subgroup.

[compare\\_signatures,HierarchicalPartition-method](#): compare signatures from different nodes.

[dimension\\_reduction,HierarchicalPartition-method](#): make dimension reduction plots.

[test\\_to\\_known\\_factors,HierarchicalPartition-method](#): test correlation between predicted subgrouping and known annotations, if available.

[cola\\_report,HierarchicalPartition-method](#): generate a HTML report for the whole analysis.

[functional\\_enrichment,HierarchicalPartition-method](#): apply functional enrichment.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

is\_best\_k-ConsensusPartition-method

*Test whether the current k is the best/optional k*

---

**Description**

Test whether the current k is the best/optional k

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> object.
k	Number of subgroups.
...	Pass to <a href="#">suggest_best_k,ConsensusPartition-method</a> .

**Details**

Optional best k is also assigned as TRUE.

**Value**

Logical scalar.

**Examples**

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

---

is\_best\_k-ConsensusPartitionList-method

*Test whether the current k is the best/optional k*

---

**Description**

Test whether the current k is the best/optional k

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartitionList-class</a> object.
k	Number of subgroups.
...	Pass to <a href="#">suggest_best_k,ConsensusPartitionList-method</a> .

**Details**

It tests on the partitions for every method.

**Value**

Logical vector.

**Examples**

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

---

is\_best\_k-dispatch      *Method dispatch page for is\_best\_k*

---

### Description

Method dispatch page for is\_best\_k.

### Dispatch

is\_best\_k can be dispatched on following classes:

- [is\\_best\\_k,ConsensusPartition-method](#), [ConsensusPartition-class](#) class method
- [is\\_best\\_k,ConsensusPartitionList-method](#), [ConsensusPartitionList-class](#) class method

### Examples

```
# no example  
NULL
```

---

is\_leaf\_node-HierarchicalPartition-method  
*Test whether a node is a leaf node*

---

### Description

Test whether a node is a leaf node

### Usage

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

### Arguments

object	A <a href="#">HierarchicalPartition-class</a> object.
node	A vector of node IDs.
merge_node	Parameters to merge sub-dendrograms, see <a href="#">merge_node_param</a> .

### Examples

```
data(golub_colo_rh)  
is_leaf_node(golub_colo_rh, all_leaves(golub_colo_rh))
```

---

is\_stable\_k-ConsensusPartition-method

*Test whether the current k corresponds to a stable partition*

---

### Description

Test whether the current k corresponds to a stable partition

### Usage

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

### Arguments

object	A <a href="#">ConsensusPartition-class</a> object.
k	Number of subgroups.
stable_PAC	Cutoff for stable PAC.
...	Pass to <a href="#">suggest_best_k,ConsensusPartition-method</a> .

### Details

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

### Value

Logical scalar.

### Examples

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

---

is\_stable\_k-ConsensusPartitionList-method

*Test whether the current k corresponds to a stable partition*

---

### Description

Test whether the current k corresponds to a stable partition

### Usage

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

### Arguments

object	A <a href="#">ConsensusPartitionList-class</a> object.
k	Number of subgroups.
...	Pass to <a href="#">suggest_best_k,ConsensusPartitionList-method</a> .

### Details

It tests on the partitions for every method.

### Value

Logical vector

### Examples

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

---

is\_stable\_k-dispatch *Method dispatch page for is\_stable\_k*

---

### Description

Method dispatch page for is\_stable\_k.

### Dispatch

is\_stable\_k can be dispatched on following classes:

- [is\\_stable\\_k,ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method
- [is\\_stable\\_k,ConsensusPartition-method, ConsensusPartition-class](#) class method

**Examples**

```
# no example
NULL
```

---

knitr_add_tab_item	<i>Add JavaScript tab in the report</i>
--------------------	---

---

**Description**

Find the knee/elbow of a list of sorted points

**Usage**

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

**Arguments**

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

**Value**

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

**Examples**

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

---

knitr_add_tab_item	<i>Add JavaScript tab in the report</i>
--------------------	---

---

**Description**

Add JavaScript tab in the report

**Usage**

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

**Arguments**

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

**Details**

Each tab contains the R source code and results generated from it (figure, tables, text, ...).  
This function is only for internal use.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[knitr\\_insert\\_tabs](#) produces a complete HTML fragment.

**Examples**

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

---

knitr\_insert\_tabs      *Generate the HTML fragment for the JavaScript tabs*

---

**Description**

Generate the HTML fragment for the JavaScript tabs

**Usage**

```
knitr_insert_tabs(uid)
```

**Arguments**

uid	A unique identifier for the div.
-----	----------------------------------

**Details**

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

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

This function is only for internal use.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

map_to_entrez_id	<i>Map to Entrez IDs</i>
------------------	--------------------------

---

**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. <code>columns(org.Hs.eg.db::org.Hs.eg.db)</code> .
org_db	The annotation database.

**Details**

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

**Value**

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

The returned object normally is used in [functional\\_enrichment](#).

### Examples

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

---

*max\_depth-HierarchicalPartition-method*  
*Max depth of the hierarchy*

---

### Description

Max depth of the hierarchy

### Usage

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

### Arguments

object            A [HierarchicalPartition-class](#) object.

### Value

A numeric value.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
data(golub_colo_rh)
max_depth(golub_colo_rh)
```

---

membership\_heatmap-ConsensusPartition-method

*Heatmap of membership in each partition*

---

### Description

Heatmap of membership in each partition

### Usage

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

### Arguments

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

### Details

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

Heatmap is split on rows by top\_n.

### Value

No value is returned.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

```
merge_node-HierarchicalPartition-method
```

*Merge node*

---

**Description**

Merge node

**Usage**

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

**Arguments**

object            A [HierarchicalPartition-class](#) object.  
node\_id            A vector of node IDs where each node is merged as a leaf node.

**Value**

A [HierarchicalPartition-class](#) object.

**Examples**

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

---

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

---

**Description**

Parameters to merge branches in subgroup dendrogram.

**Usage**

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

**Arguments**

depth                    Depth of the dendrogram.  
min\_n\_signatures        Minimal number of signatures for the partitioning on each node.  
min\_p\_signatures        Minimal fraction of signatures compared to the total number of rows on each node.

**Examples**

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

---

ncol-ConsensusPartition-method  
*Number of columns in the matrix*

---

**Description**

Number of columns in the matrix

**Usage**

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

**Arguments**

x                        A [ConsensusPartition-class](#) object.

**Examples**

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

---

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

---

**Description**

Number of columns in the matrix

**Usage**

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

**Arguments**

x                    A [ConsensusPartitionList-class](#) object.

**Examples**

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

---

ncol-dispatch                    *Method dispatch page for ncol*

---

**Description**

Method dispatch page for ncol.

**Dispatch**

ncol can be dispatched on following classes:

- [ncol,ConsensusPartitionList-method,ConsensusPartitionList-class](#) class method
- [ncol,ConsensusPartition-method,ConsensusPartition-class](#) class method
- [ncol,DownSamplingConsensusPartition-method,DownSamplingConsensusPartition-class](#) class method
- [ncol,HierarchicalPartition-method,HierarchicalPartition-class](#) class method

**Examples**

```
# no example  
NULL
```

---

ncol-DownSamplingConsensusPartition-method  
*Number of columns in the matrix*

---

**Description**

Number of columns in the matrix

**Usage**

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

**Arguments**

x                   A [DownSamplingConsensusPartition-class](#) object.

**Examples**

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

---

ncol-HierarchicalPartition-method  
*Number of columns in the matrix*

---

**Description**

Number of columns in the matrix

**Usage**

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

**Arguments**

x                   A [HierarchicalPartition-class](#) object.

**Examples**

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

---

node\_info-HierarchicalPartition-method  
*Information on the nodes*

---

**Description**

Information on the nodes

**Usage**

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

**Arguments**

object            A [HierarchicalPartition-class](#) object.

**Details**

It returns the following node-level information:

**id** Node id.

**n\_columns** Number of columns.

**n\_signatures** Number of signatures.

**p\_signatures** Percent of signatures.

**is\_leaf** Whether the node is a leaf

**Examples**

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

---

node\_level-HierarchicalPartition-method  
*Information on the nodes*

---

**Description**

Information on the nodes

**Usage**

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

**Arguments**

object            A [HierarchicalPartition-class](#) object.

**Details**

It is the same as [node\\_info,HierarchicalPartition-method](#).

**Examples**

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

---

nrow-ConsensusPartition-method  
*Number of rows in the matrix*

---

**Description**

Number of rows in the matrix

**Usage**

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

**Arguments**

x                A [ConsensusPartition-class](#) object.

**Examples**

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

---

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

---

**Description**

Number of rows in the matrix

**Usage**

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

**Arguments**

x                   A [ConsensusPartitionList-class](#) object.

**Examples**

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

---

nrow-dispatch                   *Method dispatch page for nrow*

---

**Description**

Method dispatch page for nrow.

**Dispatch**

nrow can be dispatched on following classes:

- [nrow,HierarchicalPartition-method,HierarchicalPartition-class](#) class method
- [nrow,ConsensusPartitionList-method,ConsensusPartitionList-class](#) class method
- [nrow,ConsensusPartition-method,ConsensusPartition-class](#) class method

**Examples**

```
# no example  
NULL
```

---

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

---

**Description**

Number of rows in the matrix

**Usage**

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

**Arguments**

x                    A `HierarchicalPartition-class` object.

**Examples**

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

---

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

---

**Description**

The proportion of ambiguous clustering (PAC score)

**Usage**

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

**Arguments**

consensus\_mat    A consensus matrix.  
x1                Lower bound to define "ambiguous clustering".  
x2                Upper bound to define "ambiguous clustering".  
class             Subgroup labels. If it is provided, samples with silhouette score less than the 5<sup>th</sup> percental are removed from PAC calculation.

**Details**

The PAC score is defined as  $F(x_2) - F(x_1)$  where  $F(x)$  is the CDF of the consensus matrix.

**Value**

A single numeric vaule.

**See**

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

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

---

plot\_ecdf-ConsensusPartition-method

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

---

**Description**

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

**Usage**

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

**Arguments**

object            A [ConsensusPartition-class](#) object.  
...                Other arguments.

**Details**

It plots eCDF curve for each k.

This function is mainly used in [collect\\_plots](#) and [select\\_partition\\_number](#) functions.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

See [ecdf](#) for a detailed explanation of the empirical cumulative distribution function.

**Examples**

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

---

predict\_classes-ConsensusPartition-method

*Predict classes for new samples based on cola classification*

---

**Description**

Predict classes for new samples based on cola classification

**Usage**

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

**Arguments**

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

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

## Details

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

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

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

### Value

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

### See Also

[predict\\_classes,matrix-method](#) that predicts the classes for new samples.

### Examples

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

---

predict\_classes-dispatch

*Method dispatch page for predict\_classes*

---

### Description

Method dispatch page for `predict_classes`.

### Dispatch

`predict_classes` can be dispatched on following classes:

- [predict\\_classes,matrix-method](#), [matrix-class](#) class method
- [predict\\_classes,ConsensusPartition-method](#), [ConsensusPartition-class](#) class method

**Examples**

```
# no example
NULL
```

---

```
predict_classes-matrix-method
```

*Predict classes for new samples based on signature centroid matrix*

---

**Description**

Predict classes for new samples based on signature centroid matrix

**Usage**

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

**Arguments**

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

## Details

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

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

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

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

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

## Value

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

## Examples

```
data(golub_col1)
res = golub_col1["ATC:skmeans"]
mat = get_matrix(res)
# note scaling should be applied here because the matrix was scaled in the cola analysis
mat2 = t(scale(t(mat)))

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

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

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

---

```
print.hc_table_suggest_best_k
      Print the hc_table_suggest_best_k object
```

---

**Description**

Print the hc\_table\_suggest\_best\_k object

**Usage**

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

**Arguments**

x	A hc_table_suggest_best_k object from <a href="#">suggest_best_k, HierarchicalPartition-method</a> .
...	Other arguments.

**Examples**

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

---

```
recalc_stats      Recalculate statistics in the ConsensusPartitionList object
```

---

**Description**

Recalculate statistics in the ConsensusPartitionList object

**Usage**

```
recalc_stats(r1)
```

**Arguments**

r1	A <a href="#">ConsensusPartitionList-class</a> object.
----	--

**Details**

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

**Examples**

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

---

register_NMF	<i>Register NMF partitioning method</i>
--------------	---

---

**Description**

Register NMF partitioning method

**Usage**

```
register_NMF()
```

**Details**

NMF analysis is performed by [nmf](#).

**Examples**

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

---

register_partition_methods	<i>Register user-defined partitioning methods</i>
----------------------------	---

---

**Description**

Register user-defined partitioning methods

**Usage**

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

**Arguments**

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

## Details

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

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

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

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

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

To remove a partitioning method, use `remove_partition_methods`.

There are following default partitioning methods:

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

**"kmeans"** by `kmeans`.

**"skmeans"** by `skmeans`.

**"pam"** by `pam`.

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

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

## Value

No value is returned.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## See Also

`all_partition_methods` lists all registered partitioning methods.

**Examples**

```

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

```

---

register_SOM	<i>Register SOM partitioning method</i>
--------------	---

---

**Description**

Register SOM partitioning method

**Usage**

```
register_SOM()
```

**Details**

The SOM analysis is performed by [som](#).

**Examples**

```

# There is no example
NULL

```

---

register_top_value_methods	<i>Register user-defined top-value methods</i>
----------------------------	--

---

**Description**

Register user-defined top-value methods

**Usage**

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

**Arguments**

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

## Details

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

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

Of course, you can use [rowSds](#) to give a faster calculation of row SD:

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

The registered top-value method will be used as defaults in [run\\_all\\_consensus\\_partition\\_methods](#).

To remove a top-value method, use [remove\\_top\\_value\\_methods](#).

There are four default top-value methods:

"SD" standard deviation, by [rowSds](#).

"CV" coefficient variance, calculated as  $sd/(mean+s)$  where  $s$  is the 10<sup>th</sup> 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	<i>Relabel class labels according to the reference labels</i>
---------------	---

---

### Description

Relabel class labels according to the reference labels

### Usage

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

### Arguments

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

### Details

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

Mathematically, this is called linear sum assignment problem and it is solved by [solve\\_LSAP](#).

### Value

A named vector where names correspond to the labels in class and values correspond to ref, which means  $\text{map} = \text{relabel\_class}(\text{class}, \text{ref})$ ;  $\text{map}[\text{class}]$  returns the relabelled labels.

The returned object attaches a data frame with three columns:

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

If return\_map in the [relabel\\_class](#) is set to FALSE, the function simply returns a vector of adjusted class labels.

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

**Examples**

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

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

---

remove\_partition\_methods

*Remove partitioning methods*

---

**Description**

Remove partitioning methods

**Usage**

```
remove_partition_methods(method)
```

**Arguments**

method            Name of the partitioning methods to be removed.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

remove\_top\_value\_methods  
*Remove top-value methods*

---

**Description**

Remove top-value methods

**Usage**

```
remove_top_value_methods(method)
```

**Arguments**

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

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

rownames-ConsensusPartition-method  
*Row names of the matrix*

---

**Description**

Row names of the matrix

**Usage**

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

**Arguments**

x                    A [ConsensusPartition-class](#) object.

**Examples**

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

---

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

---

**Description**

Row names of the matrix

**Usage**

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

**Arguments**

x                   A [ConsensusPartitionList-class](#) object.

**Examples**

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

---

rownames-dispatch     *Method dispatch page for rownames*

---

**Description**

Method dispatch page for rownames.

**Dispatch**

rownames can be dispatched on following classes:

- [rownames, HierarchicalPartition-method, HierarchicalPartition-class](#) class method
- [rownames, ConsensusPartition-method, ConsensusPartition-class](#) class method
- [rownames, ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method

**Examples**

```
# no example  
NULL
```

---

rownames-HierarchicalPartition-method  
*Row names of the matrix*

---

**Description**

Row names of the matrix

**Usage**

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

**Arguments**

x                   A [HierarchicalPartition-class](#) object.

**Examples**

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

---

run\_all\_consensus\_partition\_methods  
*Consensus partitioning for all combinations of methods*

---

**Description**

Consensus partitioning for all combinations of methods

**Usage**

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

**Arguments**

data	A numeric matrix where subgroups are found by columns.
top_value_method	Method which are used to extract top n rows. Allowed methods are in <a href="#">all_top_value_methods</a> and can be self-added by <a href="#">register_top_value_methods</a> .
partition_method	Method which are used to partition samples. Allowed methods are in <a href="#">all_partition_methods</a> and can be self-added by <a href="#">register_partition_methods</a> .
max_k	Maximal number of subgroups to try. The function will try 2:max_k subgroups.
k	Alternatively, you can specify a vector k.
top_n	Number of rows with top values. The value can be a vector with length > 1. When n > 5000, the function only randomly sample 5000 rows from top n rows. If top_n is a vector, partition will be applied to every values in top_n and consensus partition is summarized from all partitions.
mc.cores	Number of cores to use. This argument will be removed in future versions.
cores	Number of cores, or a cluster object returned by <a href="#">makeCluster</a> .
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 <a href="#">register_partition_methods</a> is used.
verbose	Whether to print messages.
help	Whether to print help messages.

**Details**

The function performs consensus partitioning by [consensus\\_partition](#) for all combinations of top-value methods and partitioning methods.

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

**Value**

A [ConsensusPartitionList-class](#) object. Simply type object in the interactive R session to see which functions can be applied on it.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
## Not run:
set.seed(123)
m = cbind(rbind(matrix(rnorm(20*20, mean = 1), nr = 20),
                    matrix(rnorm(20*20, mean = -1), nr = 20)),
          rbind(matrix(rnorm(20*20, mean = -1), nr = 20),
                    matrix(rnorm(20*20, mean = 1), nr = 20))
          ) + matrix(rnorm(40*40), nr = 40)
r1 = run_all_consensus_partition_methods(data = m, top_n = c(20, 30, 40))

## End(Not run)
```

---

select\_partition\_number-ConsensusPartition-method

*Several plots for determining the optimized number of subgroups*

---

**Description**

Several plots for determining the optimized number of subgroups

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> object.
mark_best	Whether mark the best k in the plot.
all_stats	Whether to show all statistics that were calculated. Used internally.

**Details**

There are following plots made:

- eCDF of the consensus matrix under each k, made by [plot\\_ecdf,ConsensusPartition-method](#),
- PAC score,
- mean silhouette score,
- the [concordance](#) for each partition to the consensus partition,
- area increase of the area under the ECDF of consensus matrix with increasing k,
- Rand index for current k compared to k - 1,
- Jaccard coefficient for current k compared to k - 1,

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

show-ConsensusPartition-method

*Print the ConsensusPartition object*

---

**Description**

Print the ConsensusPartition object

**Usage**

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

**Arguments**

object            A [ConsensusPartition-class](#) object.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

show-ConsensusPartitionList-method

*Print the ConsensusPartitionList object*

---

### Description

Print the ConsensusPartitionList object

### Usage

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

### Arguments

object            A [ConsensusPartitionList-class](#) object.

### Value

No value is returned.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

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

---

show-dispatch

*Method dispatch page for show*

---

### Description

Method dispatch page for show.

### Dispatch

show can be dispatched on following classes:

- [show, HierarchicalPartition-method, HierarchicalPartition-class](#) class method
- [show, ConsensusPartition-method, ConsensusPartition-class](#) class method
- [show, ConsensusPartitionList-method, ConsensusPartitionList-class](#) class method
- [show, DownSamplingConsensusPartition-method, DownSamplingConsensusPartition-class](#) class method

### Examples

```
# no example  
NULL
```

---

```
show-DownSamplingConsensusPartition-method  
Print the DownSamplingConsensusPartition object
```

---

### Description

Print the DownSamplingConsensusPartition object

### Usage

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

### Arguments

object            A [DownSamplingConsensusPartition-class](#) object.

### Value

No value is returned.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
data(golub_cola_ds)  
golub_cola_ds
```

---

show-HierarchicalPartition-method

*Print the HierarchicalPartition object*

---

**Description**

Print the HierarchicalPartition object

**Usage**

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

**Arguments**

object            a [HierarchicalPartition-class](#) object

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_colo_rh)  
golub_colo_rh
```

---

split\_node-HierarchicalPartition-method

*Split node*

---

**Description**

Split node

**Usage**

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

**Arguments**

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

**Details**

It applies hierarchical consensus partitioning on the specified node.

**Value**

A [HierarchicalPartition-class](#) object.

**Examples**

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

---

suggest\_best\_k-ConsensusPartition-method  
*Suggest the best number of subgroups*

---

**Description**

Suggest the best number of subgroups

**Usage**

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

**Arguments**

object	A <a href="#">ConsensusPartition-class</a> object.
jaccard_index_cutoff	The cutoff for Jaccard index for comparing to previous k.
mean_silhouette_cutoff	Cutoff for mean silhouette scores.
stable_PAC	Cutoff for stable PAC. This argument only take effect when mean_silhouette_cutoff is set to NULL.
help	Whether to print help message.

**Details**

The best k is selected according to following rules:

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

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

**Value**

The best k.

**See**

The selection of the best k can be visualized by [select\\_partition\\_number](#).

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

suggest\_best\_k-ConsensusPartitionList-method

*Suggest the best number of subgroups*

---

## Description

Suggest the best number of subgroups

## Usage

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

## Arguments

`object`            A [ConsensusPartitionList-class](#) object.  
`jaccard_index_cutoff`  
                  The cutoff for Jaccard index for comparing to previous k.

## Details

It basically gives the best k for each combination of top-value method and partitioning method by calling [suggest\\_best\\_k, ConsensusPartition-method](#).

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

## Value

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

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
data(golub_col1)  
suggest_best_k(golub_col1)
```

suggest\_best\_k-dispatch

*Method dispatch page for suggest\_best\_k*

---

### Description

Method dispatch page for suggest\_best\_k.

### Dispatch

suggest\_best\_k can be dispatched on following classes:

- [suggest\\_best\\_k,HierarchicalPartition-method,HierarchicalPartition-class](#) class method
- [suggest\\_best\\_k,ConsensusPartitionList-method,ConsensusPartitionList-class](#) class method
- [suggest\\_best\\_k,ConsensusPartition-method,ConsensusPartition-class](#) class method

### Examples

```
# no example  
NULL
```

---

suggest\_best\_k-HierarchicalPartition-method

*Guess the best number of partitions*

---

### Description

Guess the best number of partitions

### Usage

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

### Arguments

object            A [HierarchicalPartition-class](#) object.

### Details

It basically gives the best k at each node.

**Value**

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_colo_rh)
suggest_best_k(golub_colo_rh)
```

---

test\_between\_factors    *Test whether a list of factors are correlated*

---

**Description**

Test whether a list of factors are correlated

**Usage**

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

**Arguments**

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

**Details**

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

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

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

**Value**

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

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

test\_to\_known\_factors-ConsensusPartition-method

*Test correspondance between predicted subgroups and known factors*

---

**Description**

Test correspondance between predicted subgroups and known factors

**Usage**

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

**Arguments**

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

**Details**

The test is performed by [test\\_between\\_factors](#) between the predicted classes and user's annotation table.

**Value**

A data frame with the following columns:

- number of samples used to test after filtered by silhouette\_cutoff,
- p-values from the tests,
- number of subgroups.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
data(golub_col1)
res = golub_col1["ATC:skmeans"]
anno = get_anno(res)
anno
test_to_known_factors(res, k = 3)
# or explicitly specify known argument
test_to_known_factors(res, k = 3, known = anno)
```

---

test\_to\_known\_factors-ConsensusPartitionList-method

*Test correspondance between predicted classes and known factors*

---

**Description**

Test correspondance between predicted classes and known factors

**Usage**

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

**Arguments**

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

## Details

The function basically sends each [ConsensusPartition-class](#) object to [test\\_to\\_known\\_factors,ConsensusPartition-](#) and merges results afterwards.

## Value

A data frame with the following columns:

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

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

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## See Also

[test\\_between\\_factors](#), [test\\_to\\_known\\_factors](#), [ConsensusPartition-method](#)

## Examples

```
data(golub_colo)
test_to_known_factors(golub_colo)
```

---

test\_to\_known\_factors-dispatch

*Method dispatch page for test\_to\_known\_factors*

---

## Description

Method dispatch page for `test_to_known_factors`.

## Dispatch

`test_to_known_factors` can be dispatched on following classes:

- [test\\_to\\_known\\_factors,HierarchicalPartition-method,HierarchicalPartition-class](#) class method
- [test\\_to\\_known\\_factors,ConsensusPartition-method,ConsensusPartition-class](#) class method
- [test\\_to\\_known\\_factors,ConsensusPartitionList-method,ConsensusPartitionList-class](#) class method
- [test\\_to\\_known\\_factors,DownSamplingConsensusPartition-method,DownSamplingConsensusPartition-clas](#) class method

## Examples

```
# no example  
NULL
```

---

test\_to\_known\_factors-DownSamplingConsensusPartition-method  
*Test correspondance between predicted subgroups and known factors*

---

## Description

Test correspondance between predicted subgroups and known factors

## Usage

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

## Arguments

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

## Details

The test is performed by [test\\_between\\_factors](#) between the predicted classes and user's annotation table.

## Value

A data frame with the following columns:

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

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

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

---

test\_to\_known\_factors-HierarchicalPartition-method

*Test correspondance between predicted classes and known factors*

---

## Description

Test correspondance between predicted classes and known factors

## Usage

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

## Arguments

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

## Value

A data frame with columns:

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

The classifications are extracted for each depth.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

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

---

top\_elements\_overlap *Overlap of top elements from different metrics*

---

## Description

Overlap of top elements from different metrics

## Usage

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

## Arguments

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

## Details

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

## Value

No value is returned.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

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

---

top\_rows\_heatmap-ConsensusPartition-method  
*Heatmap of top rows*

---

### Description

Heatmap of top rows

### Usage

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

### Arguments

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

### Value

No value is returned.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### See Also

[top\\_rows\\_heatmap,matrix-method](#)

### Examples

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

---

top\_rows\_heatmap-ConsensusPartitionList-method

*Heatmap of top rows from different top-value methods*

---

## Description

Heatmap of top rows from different top-value methods

## Usage

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

## Arguments

object	A <a href="#">ConsensusPartitionList-class</a> 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 <a href="#">run_all_consensus_partition_methods</a> .
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.
...	Pass to <a href="#">top_rows_heatmap,matrix-method</a> .

## Value

No value is returned.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## See Also

[top\\_rows\\_heatmap,matrix-method](#)

## Examples

```
data(golub_col1)  
top_rows_heatmap(golub_col1)
```

---

top\_rows\_heatmap-dispatch

*Method dispatch page for top\_rows\_heatmap*

---

### Description

Method dispatch page for top\_rows\_heatmap.

### Dispatch

top\_rows\_heatmap can be dispatched on following classes:

- [top\\_rows\\_heatmap,ConsensusPartition-method,ConsensusPartition-class](#) class method
- [top\\_rows\\_heatmap,ConsensusPartitionList-method,ConsensusPartitionList-class](#) class method
- [top\\_rows\\_heatmap,HierarchicalPartition-method,HierarchicalPartition-class](#) class method
- [top\\_rows\\_heatmap,matrix-method,matrix-class](#) class method

### Examples

```
# no example
NULL
```

---

top\_rows\_heatmap-HierarchicalPartition-method

*Heatmap of top rows from different top-value methods*

---

### Description

Heatmap of top rows from different top-value methods

### Usage

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

**Arguments**

object	A <a href="#">HierarchicalPartition-class</a> 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 <a href="#">hierarchical_partition</a> .
anno_col	A list of colors (color is defined as a named vector) for the annotations. If anno is a data frame, anno_col should be a named list where names correspond to the column names in anno.
scale_rows	Wether to scale rows.
...	Pass to <a href="#">top_rows_heatmap,matrix-method</a>

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[top\\_rows\\_heatmap,matrix-method](#)

**Examples**

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

---

top\_rows\_heatmap-matrix-method

*Heatmap of top rows from different top-value methods*

---

**Description**

Heatmap of top rows from different top-value methods

**Usage**

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

**Arguments**

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

**Details**

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

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

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

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

---

top\_rows\_overlap-ConsensusPartitionList-method

*Overlap of top rows from different top-value methods*

---

**Description**

Overlap of top rows from different top-value methods

**Usage**

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

**Arguments**

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

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <[z.gu@dkfz.de](mailto:z.gu@dkfz.de)>

**See Also**

[top\\_elements\\_overlap](#)

**Examples**

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

---

top\_rows\_overlap-dispatch

*Method dispatch page for top\_rows\_overlap*

---

**Description**

Method dispatch page for top\_rows\_overlap.

**Dispatch**

top\_rows\_overlap can be dispatched on following classes:

- [top\\_rows\\_overlap,HierarchicalPartition-method,HierarchicalPartition-class](#) class method
- [top\\_rows\\_overlap,matrix-method,matrix-class](#) class method
- [top\\_rows\\_overlap,ConsensusPartitionList-method,ConsensusPartitionList-class](#) class method

**Examples**

```
# no example
NULL
```

---

```
top_rows_overlap-HierarchicalPartition-method
Overlap of top rows on different nodes
```

---

**Description**

Overlap of top rows on different nodes

**Usage**

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

**Arguments**

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

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[top\\_elements\\_overlap](#)

**Examples**

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

---

`top_rows_overlap-matrix-method`*Overlap of top rows from different top-value methods*

---

## Description

Overlap of top rows from different top-value methods

## Usage

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

## Arguments

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

## 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)
```

# Index

[.ConsensusPartitionList, 6  
[.HierarchicalPartition, 7  
[[.ConsensusPartitionList, 7  
[[.HierarchicalPartition, 8

adjust\_matrix, 9  
adjust\_outlier, 9, 10  
all\_leaves  
    (all\_leaves-HierarchicalPartition-method),  
    11  
all\_leaves, HierarchicalPartition-method  
    (all\_leaves-HierarchicalPartition-method),  
    11  
all\_leaves-HierarchicalPartition-method,  
    11  
all\_nodes  
    (all\_nodes-HierarchicalPartition-method),  
    11  
all\_nodes, HierarchicalPartition-method  
    (all\_nodes-HierarchicalPartition-method),  
    11  
all\_nodes-HierarchicalPartition-method,  
    11  
all\_partition\_methods, 12, 40, 43, 93, 121,  
    129  
all\_top\_value\_methods, 13, 40, 42, 93, 123,  
    129, 150, 153

aPAC, 13  
ATC, 14, 16, 38, 123  
ATC\_approx, 15, 16, 16

bicor, 15

chisq.test, 139  
cl\_consensus, 78  
cl\_dissimilarity, 24  
cl\_membership, 121  
cmdscale, 52, 54–56  
cola, 17  
cola\_opt, 17  
cola\_report (cola\_report-dispatch), 20  
cola\_report, ConsensusPartition-method,  
    44  
cola\_report, ConsensusPartition-method  
    (cola\_report-ConsensusPartition-method),  
    18  
cola\_report, ConsensusPartitionList-method,  
    45  
cola\_report, ConsensusPartitionList-method  
    (cola\_report-ConsensusPartitionList-method),  
    19  
cola\_report, HierarchicalPartition-method,  
    95  
cola\_report, HierarchicalPartition-method  
    (cola\_report-HierarchicalPartition-method),  
    21  
cola\_report-ConsensusPartition-method,  
    18  
cola\_report-ConsensusPartitionList-method,  
    19  
cola\_report-dispatch, 20  
cola\_report-HierarchicalPartition-method,  
    21  
cola\_rl, 22  
collect\_classes  
    (collect\_classes-dispatch), 25  
collect\_classes, ConsensusPartition-method,  
    44  
collect\_classes, ConsensusPartition-method  
    (collect\_classes-ConsensusPartition-method),  
    22  
collect\_classes, ConsensusPartitionList-method,  
    45  
collect\_classes, ConsensusPartitionList-method  
    (collect\_classes-ConsensusPartitionList-method),  
    23  
collect\_classes, HierarchicalPartition-method,  
    94  
collect\_classes, HierarchicalPartition-method

- (collect\_classes-HierarchicalPartition-method),  
25
- (collect\_classes-DownSamplingConsensusPartition-method),  
32
- collect\_classes-ConsensusPartition-method,  
22
- colnames,HierarchicalPartition-method  
(colnames-HierarchicalPartition-method),  
33
- collect\_classes-ConsensusPartitionList-method,  
23
- colnames-ConsensusPartition-method, 31
- collect\_classes-dispatch, 25
- colnames-ConsensusPartitionList-method,  
31
- collect\_classes-HierarchicalPartition-method,  
25
- colnames-dispatch, 32
- collect\_plots, 113
- colnames-DownSamplingConsensusPartition-method,  
32
- collect\_plots (collect\_plots-dispatch),  
28
- colnames-HierarchicalPartition-method,  
33
- collect\_plots,ConsensusPartition-method,  
44
- colorRamp2, 115, 117
- collect\_plots,ConsensusPartition-method  
(collect\_plots-ConsensusPartition-method),  
26
- compare\_partitions  
(compare\_partitions-ConsensusPartition-method),  
33
- collect\_plots,ConsensusPartitionList-method,  
45
- compare\_partitions,ConsensusPartition-method  
(compare\_partitions-ConsensusPartition-method),  
33
- collect\_plots,ConsensusPartitionList-method  
(collect\_plots-ConsensusPartitionList-method),  
27
- compare\_partitions-ConsensusPartition-method,  
33
- collect\_plots-ConsensusPartition-method,  
26
- compare\_signatures  
(compare\_signatures-dispatch),  
35
- collect\_plots-ConsensusPartitionList-method,  
27
- compare\_signatures,ConsensusPartition-method  
(compare\_signatures-ConsensusPartition-method),  
34
- collect\_plots-dispatch, 28
- compare\_signatures,HierarchicalPartition-method,  
95
- collect\_stats (collect\_stats-dispatch),  
30
- compare\_signatures,HierarchicalPartition-method  
(compare\_signatures-HierarchicalPartition-method),  
36
- collect\_stats,ConsensusPartition-method  
(collect\_stats-ConsensusPartition-method),  
29
- compare\_signatures-ConsensusPartition-method,  
34
- collect\_stats,ConsensusPartitionList-method  
(collect\_stats-ConsensusPartitionList-method),  
29
- compare\_signatures-dispatch, 35
- collect\_stats-ConsensusPartition-method,  
29
- compare\_signatures-HierarchicalPartition-method,  
36
- collect\_stats-ConsensusPartitionList-method,  
29
- concordance, 36, 87, 130
- collect\_stats-dispatch, 30
- config\_ATC, 37
- colnames (colnames-dispatch), 32
- consensus\_heatmap, 27
- colnames,ConsensusPartition-method  
(colnames-ConsensusPartition-method),  
31
- consensus\_heatmap  
(consensus\_heatmap-ConsensusPartition-method),  
38
- colnames,ConsensusPartitionList-method  
(colnames-ConsensusPartitionList-method),  
31
- consensus\_heatmap,ConsensusPartition-method,  
44
- colnames,DownSamplingConsensusPartition-method  
(consensus\_heatmap-ConsensusPartition-method),

- 38
- consensus\_heatmap-ConsensusPartition-method, 38
- consensus\_partition, 23, 38, 40, 43, 44, 64, 82, 93, 104, 120, 121, 129, 140, 141
- consensus\_partition\_by\_down\_sampling, 42, 57, 66, 143
- ConsensusPartition, 34
- ConsensusPartition (ConsensusPartition-class), 44
- ConsensusPartition-class, 44
- ConsensusPartitionList (ConsensusPartitionList-class), 45
- ConsensusPartitionList-class, 45
- cor.test, 118, 139
- correspond\_between\_rankings, 46, 48, 145, 151–153
- correspond\_between\_two\_rankings, 46, 47
- cutree, 121
- david\_enrichment, 48
- dim.ConsensusPartition, 50
- dim.ConsensusPartitionList, 50
- dim.DownSamplingConsensusPartition, 51
- dim.HierarchicalPartition, 51
- dimension\_reduction, 27
- dimension\_reduction (dimension\_reduction-dispatch), 53
- dimension\_reduction,ConsensusPartition-method, 44
- dimension\_reduction,ConsensusPartition-method (dimension\_reduction-ConsensusPartition-method), 52
- dimension\_reduction,DownSamplingConsensusPartition-method (dimension\_reduction-DownSamplingConsensusPartition-method), 53
- dimension\_reduction,HierarchicalPartition-method, 95
- dimension\_reduction,HierarchicalPartition-method (dimension\_reduction-HierarchicalPartition-method), 54
- dimension\_reduction,matrix-method (dimension\_reduction-matrix-method), 55
- dimension\_reduction-ConsensusPartition-method, 52
- dimension\_reduction-dispatch, 53
- dimension\_reduction-DownSamplingConsensusPartition-method, 53
- dimension\_reduction-HierarchicalPartition-method, 54
- dimension\_reduction-matrix-method, 55
- DownSamplingConsensusPartition (DownSamplingConsensusPartition-class), 57
- DownSamplingConsensusPartition-class, 57
- ecdf, 114
- enrichDO, 59
- enricher, 59
- enrichGO, 59
- enrichKEGG, 59
- enrichPathway, 59
- euler, 145, 151–153
- Extract.ConsensusPartitionList ([.ConsensusPartitionList), 6
- Extract.HierarchicalPartition ([.HierarchicalPartition), 7
- ExtractExtract.ConsensusPartitionList ([[.ConsensusPartitionList), 7
- ExtractExtract.HierarchicalPartition ([[.HierarchicalPartition), 8
- FALSE, 124
- FCC, 57
- find\_best\_km, 58
- functional\_enrichment, 49, 102
- functional\_enrichment (functional\_enrichment-dispatch), 60
- functional\_enrichment,ANY-method (functional\_enrichment-ANY-method), 59
- functional\_enrichment,ConsensusPartition-method, 44
- functional\_enrichment,ConsensusPartition-method (functional\_enrichment-ConsensusPartition-method), 50
- functional\_enrichment,ConsensusPartitionList-method, 45
- functional\_enrichment,ConsensusPartitionList-method (functional\_enrichment-ConsensusPartitionList-method), 61
- functional\_enrichment,HierarchicalPartition-method, 95

functional\_enrichment, HierarchicalPartition-method, anno\_col-HierarchicalPartition-method,  
 (functional\_enrichment-HierarchicalPartition-method),  
 63 get\_children\_nodes  
 functional\_enrichment-ANY-method, 59 (get\_children\_nodes-HierarchicalPartition-method),  
 functional\_enrichment-ConsensusPartition-method, 70  
 60 get\_children\_nodes, HierarchicalPartition-method  
 functional\_enrichment-ConsensusPartitionList-method, (get\_children\_nodes-HierarchicalPartition-method),  
 61 70  
 functional\_enrichment-dispatch, 62 get\_children\_nodes-HierarchicalPartition-method,  
 functional\_enrichment-HierarchicalPartition-method, 70  
 63 get\_classes (get\_classes-dispatch), 72  
 get\_classes, ConsensusPartition-method,  
 44  
 get\_anno (get\_anno-dispatch), 65  
 get\_anno, ConsensusPartition-method get\_classes, ConsensusPartition-method  
 (get\_anno-ConsensusPartition-method), (get\_classes-ConsensusPartition-method),  
 64 70  
 get\_anno, ConsensusPartitionList-method get\_classes, ConsensusPartitionList-method,  
 (get\_anno-ConsensusPartitionList-method), 45  
 65 get\_classes, ConsensusPartitionList-method  
 get\_anno, DownSamplingConsensusPartition-method (get\_classes-ConsensusPartitionList-method),  
 (get\_anno-DownSamplingConsensusPartition-method), 71  
 66 get\_classes, DownSamplingConsensusPartition-method  
 get\_anno, HierarchicalPartition-method (get\_classes-DownSamplingConsensusPartition-method),  
 (get\_anno-HierarchicalPartition-method), 72  
 67 get\_classes, HierarchicalPartition-method,  
 94  
 get\_anno-ConsensusPartition-method, 64  
 get\_anno-ConsensusPartitionList-method, get\_classes, HierarchicalPartition-method  
 65 (get\_classes-HierarchicalPartition-method),  
 73  
 get\_anno-dispatch, 65  
 get\_anno-DownSamplingConsensusPartition-method, get\_classes-ConsensusPartition-method,  
 66 70  
 get\_anno-HierarchicalPartition-method, get\_classes-ConsensusPartitionList-method,  
 67 71  
 get\_anno\_col (get\_anno\_col-dispatch), 69 get\_classes-dispatch, 72  
 get\_anno\_col, ConsensusPartition-method get\_classes-DownSamplingConsensusPartition-method,  
 (get\_anno\_col-ConsensusPartition-method), 72  
 67 get\_classes-HierarchicalPartition-method,  
 73  
 get\_anno\_col, ConsensusPartitionList-method (get\_anno\_col-ConsensusPartitionList-method),  
 68 get\_consensus  
 (get\_consensus-ConsensusPartition-method),  
 74  
 get\_anno\_col, HierarchicalPartition-method (get\_anno\_col-ConsensusPartitionList-method),  
 (get\_anno\_col-HierarchicalPartition-method), 44  
 69  
 get\_anno\_col-ConsensusPartition-method, get\_consensus, ConsensusPartition-method  
 67 (get\_consensus-ConsensusPartition-method),  
 74  
 get\_anno\_col-ConsensusPartitionList-method, 74  
 68 get\_consensus-ConsensusPartition-method,  
 74  
 get\_anno\_col-dispatch, 69

- get\_matrix (get\_matrix-dispatch), 76
- get\_matrix, ConsensusPartition-method, 44, 45
- get\_matrix, ConsensusPartition-method (get\_matrix-ConsensusPartition-method), 75
- get\_matrix, ConsensusPartitionList-method (get\_matrix-ConsensusPartitionList-method), 75
- get\_matrix, DownSamplingConsensusPartition-method (get\_matrix-DownSamplingConsensusPartition-method), 77
- get\_matrix, HierarchicalPartition-method, 95
- get\_matrix, HierarchicalPartition-method (get\_matrix-HierarchicalPartition-method), 77
- get\_matrix-ConsensusPartition-method, 75
- get\_matrix-ConsensusPartitionList-method, 75
- get\_matrix-dispatch, 76
- get\_matrix-DownSamplingConsensusPartition-method, 77
- get\_matrix-HierarchicalPartition-method, 77
- get\_membership (get\_membership-dispatch), 80
- get\_membership, ConsensusPartition-method, 44
- get\_membership, ConsensusPartition-method (get\_membership-ConsensusPartition-method), 78
- get\_membership, ConsensusPartitionList-method, 45
- get\_membership, ConsensusPartitionList-method (get\_membership-ConsensusPartitionList-method), 79
- get\_membership-ConsensusPartition-method, 78
- get\_membership-ConsensusPartitionList-method, 79
- get\_membership-dispatch, 80
- get\_param (get\_param-ConsensusPartition-method), 80
- get\_param, ConsensusPartition-method (get\_param-ConsensusPartition-method), 80
- get\_param-ConsensusPartition-method, 80
- get\_signatures, 27
- get\_signatures (get\_signatures-dispatch), 84
- get\_signatures, ConsensusPartition-method, 44
- get\_signatures, ConsensusPartition-method (get\_signatures-ConsensusPartition-method), 81
- get\_signatures, DownSamplingConsensusPartition-method (get\_signatures-DownSamplingConsensusPartition-method), 85
- get\_signatures, HierarchicalPartition-method, 95
- get\_signatures, HierarchicalPartition-method (get\_signatures-HierarchicalPartition-method), 85
- get\_signatures-ConsensusPartition-method, 81
- get\_signatures-dispatch, 84
- get\_signatures-DownSamplingConsensusPartition-method, 85
- get\_signatures-HierarchicalPartition-method, 85
- get\_stats (get\_stats-dispatch), 89
- get\_stats, ConsensusPartition-method, 44
- get\_stats, ConsensusPartition-method (get\_stats-ConsensusPartition-method), 87
- get\_stats, ConsensusPartitionList-method, 45
- get\_stats, ConsensusPartitionList-method (get\_stats-ConsensusPartitionList-method), 88
- get\_stats-ConsensusPartition-method, 87
- get\_stats-ConsensusPartitionList-method, 88
- get\_stats-dispatch, 89
- golub\_col, 89
- golub\_col, ds, 91
- golub\_col, rh, 91
- Heatmap, 150

- [hierarchical\\_partition](#), [26](#), [63](#), [67](#), [86](#), [92](#),  
[94](#), [135](#), [144](#), [149](#)  
[HierarchicalPartition](#), [83](#)  
[HierarchicalPartition](#)  
 (HierarchicalPartition-class),  
[94](#)  
[HierarchicalPartition-class](#), [94](#)  
[impute.knn](#), [9](#)  
[is\\_best\\_k](#) ([is\\_best\\_k-dispatch](#)), [97](#)  
[is\\_best\\_k](#), [ConsensusPartition-method](#)  
 ([is\\_best\\_k-ConsensusPartition-method](#)), [membership\\_heatmap](#)  
[95](#)  
[is\\_best\\_k](#), [ConsensusPartitionList-method](#)  
 ([is\\_best\\_k-ConsensusPartitionList-method](#)), [membership\\_heatmap](#), [ConsensusPartition-method](#),  
[96](#)  
[is\\_best\\_k-ConsensusPartition-method](#),  
[95](#)  
[is\\_best\\_k-ConsensusPartitionList-method](#),  
[96](#)  
[is\\_best\\_k-dispatch](#), [97](#)  
[is\\_leaf\\_node](#)  
 ([is\\_leaf\\_node-HierarchicalPartition-method](#)), [merge\\_node](#), [HierarchicalPartition-method](#),  
[97](#)  
[is\\_leaf\\_node](#), [HierarchicalPartition-method](#)  
 ([is\\_leaf\\_node-HierarchicalPartition-method](#)), [merge\\_node](#), [HierarchicalPartition-method](#),  
[97](#)  
[is\\_leaf\\_node-HierarchicalPartition-method](#),  
[97](#)  
[is\\_stable\\_k](#) ([is\\_stable\\_k-dispatch](#)), [99](#)  
[is\\_stable\\_k](#), [ConsensusPartition-method](#)  
 ([is\\_stable\\_k-ConsensusPartition-method](#)), [merge\\_node\\_param](#), [11](#), [12](#), [26](#), [36](#), [55](#), [63](#), [70](#),  
[98](#)  
[is\\_stable\\_k](#), [ConsensusPartitionList-method](#)  
 ([is\\_stable\\_k-ConsensusPartitionList-method](#)), [ncol](#) ([ncol-dispatch](#)), [107](#)  
[99](#)  
[is\\_stable\\_k-ConsensusPartition-method](#),  
[98](#)  
[is\\_stable\\_k-ConsensusPartitionList-method](#),  
[99](#)  
[is\\_stable\\_k-dispatch](#), [99](#)  
[kmeans](#), [121](#)  
[knee\\_finder2](#), [100](#)  
[knitr\\_add\\_tab\\_item](#), [100](#), [102](#)  
[knitr\\_insert\\_tabs](#), [101](#), [101](#), [102](#)  
[makeCluster](#), [19](#), [21](#), [28](#), [41](#), [43](#), [93](#), [115](#), [117](#),  
[129](#)  
[map\\_to\\_entrez\\_id](#), [102](#)  
[max\\_depth](#)  
 ([max\\_depth-HierarchicalPartition-method](#)),  
[103](#)  
[max\\_depth](#), [HierarchicalPartition-method](#)  
 ([max\\_depth-HierarchicalPartition-method](#)),  
[103](#)  
[max\\_depth-HierarchicalPartition-method](#),  
[103](#)  
[Mclust](#), [121](#)  
[membership\\_heatmap](#), [27](#)  
 ([membership\\_heatmap-ConsensusPartition-method](#)),  
[104](#)  
[membership\\_heatmap](#), [ConsensusPartition-method](#),  
[44](#)  
[membership\\_heatmap](#), [ConsensusPartition-method](#)  
 ([membership\\_heatmap-ConsensusPartition-method](#)),  
[104](#)  
[membership\\_heatmap-ConsensusPartition-method](#),  
[104](#)  
[merge\\_node](#)  
 ([merge\\_node-HierarchicalPartition-method](#)), [105](#)  
[merge\\_node](#), [HierarchicalPartition-method](#)  
 ([merge\\_node-HierarchicalPartition-method](#)),  
[105](#)  
[merge\\_node-HierarchicalPartition-method](#),  
[105](#)  
[merge\\_node\\_param](#), [11](#), [12](#), [26](#), [36](#), [55](#), [63](#), [70](#),  
[73](#), [86](#), [97](#), [105](#), [144](#)  
[ncol](#) ([ncol-dispatch](#)), [107](#)  
[ncol](#), [ConsensusPartition-method](#)  
 ([ncol-ConsensusPartition-method](#)),  
[106](#)  
[ncol](#), [ConsensusPartitionList-method](#)  
 ([ncol-ConsensusPartitionList-method](#)),  
[107](#)  
[ncol](#), [DownSamplingConsensusPartition-method](#)  
 ([ncol-DownSamplingConsensusPartition-method](#)),  
[108](#)  
[ncol](#), [HierarchicalPartition-method](#)  
 ([ncol-HierarchicalPartition-method](#)),  
[108](#)  
[ncol-ConsensusPartition-method](#), [106](#)  
[ncol-ConsensusPartitionList-method](#),  
[107](#)  
[ncol-dispatch](#), [107](#)

- ncol-DownSamplingConsensusPartition-method, [108](#)
- ncol-HierarchicalPartition-method, [108](#)
- nmf, [120](#)
- node\_info
  - (node\_info-HierarchicalPartition-method), [109](#)
- node\_info,HierarchicalPartition-method
  - (node\_info-HierarchicalPartition-method), [109](#)
- node\_info-HierarchicalPartition-method, [109](#)
- node\_level
  - (node\_level-HierarchicalPartition-method), [109](#)
- node\_level,HierarchicalPartition-method
  - (node\_level-HierarchicalPartition-method), [109](#)
- node\_level-HierarchicalPartition-method, [109](#)
- nrow (nrow-dispatch), [111](#)
- nrow,ConsensusPartition-method
  - (nrow-ConsensusPartition-method), [110](#)
- nrow,ConsensusPartitionList-method
  - (nrow-ConsensusPartitionList-method), [111](#)
- nrow,HierarchicalPartition-method
  - (nrow-HierarchicalPartition-method), [112](#)
- nrow-ConsensusPartition-method, [110](#)
- nrow-ConsensusPartitionList-method, [111](#)
- nrow-dispatch, [111](#)
- nrow-HierarchicalPartition-method, [112](#)
  
- oneway.test, [139](#)
  
- PAC, [87](#), [112](#), [130](#)
- pam, [121](#)
- plot.euler, [145](#), [151–153](#)
- plot\_ecdf, [27](#)
- plot\_ecdf
  - (plot\_ecdf-ConsensusPartition-method), [113](#)
- plot\_ecdf,ConsensusPartition-method
  - (plot\_ecdf-ConsensusPartition-method), [113](#)
- plot\_ecdf-ConsensusPartition-method, [113](#)
- prcomp, [52](#), [54–56](#)
- predict\_classes
  - (predict\_classes-dispatch), [116](#)
  - (predict\_classes,ConsensusPartition-method), [114](#)
  - (predict\_classes-ConsensusPartition-method), [114](#)
  - (predict\_classes,matrix-method), [117](#)
  - (predict\_classes-matrix-method), [114](#)
- predict\_classes-ConsensusPartition-method, [114](#)
- predict\_classes-dispatch, [116](#)
- predict\_classes-matrix-method, [117](#)
- print.hc\_table\_suggest\_best\_k, [119](#)
- recalc\_stats, [119](#)
- register\_NMF, [120](#), [121](#)
- register\_partition\_methods, [12](#), [40](#), [41](#), [43](#), [120](#), [129](#)
- register\_SOM, [121](#), [122](#)
- register\_top\_value\_methods, [13](#), [40](#), [42](#), [122](#), [129](#)
- relabel\_class, [37](#), [124](#), [124](#)
- remove\_partition\_methods, [121](#), [125](#)
- remove\_top\_value\_methods, [123](#), [126](#)
- rowMads, [123](#)
- rownames (rownames-dispatch), [127](#)
- rownames,ConsensusPartition-method
  - (rownames-ConsensusPartition-method), [126](#)
- rownames,ConsensusPartitionList-method
  - (rownames-ConsensusPartitionList-method), [127](#)
- rownames,HierarchicalPartition-method
  - (rownames-HierarchicalPartition-method), [128](#)
- rownames-ConsensusPartition-method, [126](#)
- rownames-ConsensusPartitionList-method, [127](#)
- rownames-dispatch, [127](#)
- rownames-HierarchicalPartition-method, [128](#)
- rowSds, [123](#)
- Rtsne, [52](#), [54–56](#)
- run\_all\_consensus\_partition\_methods, [23](#), [24](#), [27](#), [38](#), [41](#), [45](#), [60](#), [61](#), [64](#), [65](#),

- 82, 104, 120, 121, 123, 128, 140, 141, 147
- scale, 120
- select\_partition\_number, 113, 136
- select\_partition\_number  
 (select\_partition\_number-ConsensusPartition-method), 130
- select\_partition\_number, ConsensusPartition-method, 44
- select\_partition\_number, ConsensusPartition-method  
 (select\_partition\_number-ConsensusPartition-method), 130
- select\_partition\_number-ConsensusPartition-method, 130
- show (show-dispatch), 132
- show, ConsensusPartition-method  
 (show-ConsensusPartition-method), 131
- show, ConsensusPartitionList-method  
 (show-ConsensusPartitionList-method), 132
- show, DownSamplingConsensusPartition-method  
 (show-DownSamplingConsensusPartition-method), 133
- show, HierarchicalPartition-method  
 (show-HierarchicalPartition-method), 134
- show-ConsensusPartition-method, 131
- show-ConsensusPartitionList-method, 132
- show-dispatch, 132
- show-DownSamplingConsensusPartition-method, 133
- show-HierarchicalPartition-method, 134
- skmeans, 121
- solve\_LSAP, 124
- som, 122
- split\_node  
 (split\_node-HierarchicalPartition-method), 134
- split\_node, HierarchicalPartition-method  
 (split\_node-HierarchicalPartition-method), 134
- split\_node-HierarchicalPartition-method, 134
- suggest\_best\_k  
 (suggest\_best\_k-dispatch), 138
- suggest\_best\_k, ConsensusPartition-method, 44
- suggest\_best\_k, ConsensusPartition-method  
 (suggest\_best\_k-ConsensusPartition-method), 135
- suggest\_best\_k, ConsensusPartitionList-method,  
 (suggest\_best\_k-ConsensusPartitionList-method), 137
- suggest\_best\_k, HierarchicalPartition-method,  
 (suggest\_best\_k-HierarchicalPartition-method), 95
- suggest\_best\_k-ConsensusPartition-method, 135
- suggest\_best\_k-ConsensusPartitionList-method, 137
- suggest\_best\_k-dispatch, 138
- suggest\_best\_k-HierarchicalPartition-method, 138
- test\_between\_factors, 139, 140, 142, 143
- test\_to\_known\_factors  
 (test\_to\_known\_factors-dispatch), 142
- test\_to\_known\_factors, ConsensusPartition-method, 44
- test\_to\_known\_factors, ConsensusPartition-method  
 (test\_to\_known\_factors-ConsensusPartition-method), 140
- test\_to\_known\_factors, ConsensusPartitionList-method, 45
- test\_to\_known\_factors, ConsensusPartitionList-method  
 (test\_to\_known\_factors-ConsensusPartitionList-method), 141
- test\_to\_known\_factors, DownSamplingConsensusPartition-method,  
 (test\_to\_known\_factors-DownSamplingConsensusPartition-method), 143
- test\_to\_known\_factors, HierarchicalPartition-method,  
 (test\_to\_known\_factors-HierarchicalPartition-method), 95
- test\_to\_known\_factors, HierarchicalPartition-method  
 (test\_to\_known\_factors-HierarchicalPartition-method), 144
- test\_to\_known\_factors-ConsensusPartition-method, 140
- test\_to\_known\_factors-ConsensusPartitionList-method, 141

- test\_to\_known\_factors-dispatch, [142](#)
- test\_to\_known\_factors-DownSamplingConsensusPartition-method, [143](#)
- test\_to\_known\_factors-HierarchicalPartition-method, [144](#)
- top\_elements\_overlap, [145](#), [151–153](#)
- top\_rows\_heatmap
  - (top\_rows\_heatmap-dispatch), [148](#)
- top\_rows\_heatmap,ConsensusPartition-method
  - (top\_rows\_heatmap-ConsensusPartition-method), [146](#)
- top\_rows\_heatmap,ConsensusPartitionList-method, [45](#)
- top\_rows\_heatmap,ConsensusPartitionList-method
  - (top\_rows\_heatmap-ConsensusPartitionList-method), [147](#)
- top\_rows\_heatmap,HierarchicalPartition-method
  - (top\_rows\_heatmap-HierarchicalPartition-method), [148](#)
- top\_rows\_heatmap,matrix-method
  - (top\_rows\_heatmap-matrix-method), [149](#)
- top\_rows\_heatmap-ConsensusPartition-method, [146](#)
- top\_rows\_heatmap-ConsensusPartitionList-method, [147](#)
- top\_rows\_heatmap-dispatch, [148](#)
- top\_rows\_heatmap-HierarchicalPartition-method, [148](#)
- top\_rows\_heatmap-matrix-method, [149](#)
- top\_rows\_overlap
  - (top\_rows\_overlap-dispatch), [151](#)
- top\_rows\_overlap,ConsensusPartitionList-method, [45](#)
- top\_rows\_overlap,ConsensusPartitionList-method
  - (top\_rows\_overlap-ConsensusPartitionList-method), [150](#)
- top\_rows\_overlap,HierarchicalPartition-method
  - (top\_rows\_overlap-HierarchicalPartition-method), [152](#)
- top\_rows\_overlap,matrix-method
  - (top\_rows\_overlap-matrix-method), [153](#)
- top\_rows\_overlap-ConsensusPartitionList-method, [150](#)
- top\_rows\_overlap-dispatch, [151](#)
- top\_rows\_overlap-HierarchicalPartition-method, [151](#)
- top\_rows\_overlap-matrix-method, [153](#)
- umap, [52](#), [54–56](#)
- unique, [81](#)
- unit, [47](#)
- UpSet, [145](#), [151–153](#)
- venn, [145](#), [151–153](#)