# Package 'CEMiTool'

April 15, 2019

**Title** Co-expression Modules identification Tool

**Version** 1.6.11

**Description** The CEMiTool package unifies the discovery and the analysis of coexpression gene modules in a fully automatic manner, while providing a user-friendly html report with high quality graphs. Our tool evaluates if modules contain genes that are over-represented by specific pathways or that are altered in a specific sample group. Additionally, CEMiTool is able to integrate transcriptomic data with interactome information, identifying the potential hubs on each network.

**Depends** R (>= 3.5)

**Imports** methods, scales, gRbase, dplyr, data.table (>= 1.9.4), WGCNA, grid, ggplot2, ggpmisc, ggthemes, ggrepel, sna, clusterProfiler, fgsea, stringr, knitr, rmarkdown, igraph, DT, htmltools, pracma, intergraph, grDevices, utils, network, matrixStats, ggdendro, gridExtra, gtable, GeneOverlap, limma, tidyr, plyr, ff, ffbase, RColorBrewer

**Suggests** testthat, BiocManager

**License** GPL-3

**Encoding** UTF-8

**biocViews** GeneExpression, Transcriptomics, GraphAndNetwork, mRNAMicroarray, RNASeq, Network, NetworkEnrichment, Pathways, ImmunoOncology

**LazyData** true

**RoxygenNote** 6.1.0

**VignetteBuilder** knitr

**git_url** https://git.bioconductor.org/packages/CEMiTool

**git_branch** RELEASE_3_8

**git_last_commit** d4c4ae4

**git_last_commit_date** 2019-03-15

**Date/Publication** 2019-04-15

**Author** Pedro Russo [aut],
Gustavo Ferreira [aut],
Matheus Bürger [aut],
Lucas Cardozo [aut],
Diogenes Lima [aut],

Thiago Hirata [aut],
Melissa Lever [aut],
Helder Nakaya [aut, cre]

**Maintainer** Helder Nakaya <hnakaya@usp.br>

# R topics documented:

---

`adj_data` *Get or set adjacency matrix value*

---

## Description

This function takes a `CEMiTool` object containing expression values and returns a CEMiTool object with an adjacency matrix in the adjacency slot.

## Usage

```
adj_data(cem, ...)

## S4 method for signature 'CEMiTool'
adj_data(cem)

adj_data(cem) <- value

## S4 replacement method for signature 'CEMiTool'
adj_data(cem) <- value
```

## Arguments

| | |
|---|---|
| `cem` | Object of class `CEMiTool` |
| `...` | Optional parameters. |
| `value` | Object of class `matrix` containing adjacency data. Only used for setting adjacency values to CEMiTool object. |

## Value

Object of class `matrix` with adjacency values or object of class `CEMiTool`.

## Examples

```
# Get example expression data
data(expr0)
# Initialize new CEMiTool object with expression
cem <- new_cem(expr0, filter=TRUE, apply_vst=FALSE)
# Calculate adjacency matrix with example beta value 8
cem <- get_adj(cem, beta=8)
# Return adjacency matrix
adj <- adj_data(cem)
# Check result
adj[1:5, 1:5]
# Set adjacency matrix to CEMiTool object
adj_data(cem) <- adj
```

---

# cem *CEMiTool Object*

---

## Description

This object can be used as input for CEMiTool functions. Data used are from `expr` and `sample_annot`.

## Usage

```
data(cem)
```

## Format

An object of class `CEMiTool`

## Examples

```
# Get example CEMiTool object
data(cem)
cem
```

---

# cemitool *Full gene co-expression analysis*

---

## Description

Defines co-expression modules and runs several different analyses.

## Usage

```
cemitool(expr, annot, gmt, interactions, filter = TRUE,
  filter_pval = 0.1, apply_vst = FALSE, n_genes, eps = 0.1,
  cor_method = c("pearson", "spearman"), cor_function = "cor",
  network_type = "unsigned", tom_type = "signed", set_beta = NULL,
  force_beta = FALSE, sample_name_column = "SampleName",
  class_column = "Class", merge_similar = TRUE, rank_method = "mean",
  ora_pval = 0.05, gsea_scale = TRUE, gsea_min_size = 15,
  gsea_max_size = 1000, min_ngen = 30, diss_thresh = 0.8,
  plot = TRUE, plot_diagnostics = TRUE, order_by_class = TRUE,
  center_func = "mean", directed = FALSE, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| expr | Gene expression data.frame. |
| annot | Sample annotation data.frame. |
| gmt | A data.frame containing two columns, one with pathways and one with genes |
| interactions | A data.frame containing two columns with gene names. |
| filter | Logical. If TRUE, will filter expression data. |
| filter_pval | P-value threshold for filtering. Default 0.1. |
| apply_vst | Logical. If TRUE, will apply Variance Stabilizing Transform before filtering genes. Currently ignored if parameter filter is FALSE. |
| n_genes | Number of genes left after filtering. |
| eps | A value for accepted R-squared interval between subsequent beta values. Default is 0.1. |
| cor_method | A character string indicating which correlation coefficient is to be computed. One of "pearson" or "spearman". Default is "pearson". |
| cor_function | A character string indicating the correlation function to be used. Supported functions are currently 'cor' and 'bicor'. Default is "cor" |
| network_type | A character string indicating if network type should be computed as "signed" or "unsigned". Default is "unsigned" |
| tom_type | A character string indicating if the TOM type should be computed as "signed" or "unsigned". Default is "signed" |
| set_beta | A value to override the automatically selected beta value. Default is NULL. |
| force_beta | Whether or not to automatically force a beta value based on number of samples. Default is FALSE. |
| sample_name_column | A character string indicating the sample column name of the annotation table. |
| class_column | A character string indicating the class column name of the annotation table. |
| merge_similar | Logical. If TRUE, merge similar modules. |
| rank_method | Character string indicating how to rank genes. Either "mean" (the default) or "median". |
| ora_pval | P-value for overrepresentation analysis. Default 0.05. |
| gsea_scale | If TRUE, apply z-score transformation for GSEA analysis. Default is TRUE |
| gsea_min_size | Minimum size of gene sets for GSEA analysis. Default is 15 |
| gsea_max_size | Maximum size of gene sets for GSEA analysis. Default is 1000 |

| | |
|---|---|
| min_ngen | Minimal number of genes per submodule. Default 30. |
| diss_thresh | Module merging correlation threshold for eigengene similarity. Default 0.8. |
| plot | Logical. If TRUE, plots all figures. |
| plot_diagnostics | Logical. If TRUE, creates diagnostic plots. Overwritten if plot=FALSE. |
| order_by_class | Logical. If TRUE, samples in profile plot are ordered by the groups defined by the class_column slot in the sample annotation file. Ignored if there is no sample_annotation file. Default TRUE. |
| center_func | Character string indicating the centrality measure to show in the plot. Either 'mean' (the default) or 'median'. |
| directed | Logical. If TRUE, the igraph objects in interactions slot will be directed. |
| verbose | Logical. If TRUE, reports analysis steps. |

## Value

Object of class CEMiTool

## Examples

```
# Get example expression data
data(expr0)
# Run CEMiTool analyses
cem <- cemitool(expr=expr0)
# Run CEMiTool applying Variance Stabilizing Transformation to data
cem <- cemitool(expr=expr0, apply_vst=TRUE)
# Run CEMiTool with additional processing messages
cem <- cemitool(expr=expr0, verbose=TRUE)

## Not run:
# Run full CEMiTool analysis
## Get example sample annotation data
data(sample_annot)
## Read example pathways file
gmt_fname <- system.file("extdata", "pathways.gmt", package = "CEMiTool")
gmt_in <- read_gmt(gmt_fname)
## Get example interactions file
int_df <- read.delim(system.file("extdata", "interactions.tsv", package = "CEMiTool"))
## Run CEMiTool
cem <- cemitool(expr=expr0, annot=sample_annot, gmt=gmt_in,
    interactions=int_df, verbose=TRUE, plot=TRUE)

# Create report as html file
generate_report(cem, directory = "./Report", output_format="html_document")

# Write analysis results into files
write_files(cem, directory="./Tables", force=TRUE)

# Save all plots
save_plots(cem, "all", directory="./Plots")

## End(Not run)
```

CEMiTool-class *An S4 class to represent the CEMiTool analysis.*

## Description

An S4 class to represent the CEMiTool analysis.

## Slots

`expression` Gene expression `data.frame`.

`sample_annotation` Sample annotation `data.frame`.

`fit_indices` `data.frame` containing scale-free model fit, soft-threshold and network parameters.

`selected_genes` Character `vector` containing the names of genes selected for analysis

`module` Genes in modules information `data.frame`.

`enrichment` `list` with modules enrichment results for sample classes.

`ora` Over-representation analysis results `data.frame`.

`interactions` `list` containing gene interactions present in modules.

`interaction_plot` list of ggplot graphs with module gene interactions.

`profile_plot` list of ggplot graphs with gene expression profile per module.

`enrichment_plot` ggplot graph for enrichment analysis results.

`beta_r2_plot` ggplot graph with scale-free topology fit results for each soft-threshold.

`mean_k_plot` ggplot graph with mean network connectivity.

`barplot_ora` list of ggplot graphs with over-representation analysis results per module.

`sample_tree_plot` gtable containing sample dendrogram with class labels and clinical data (if available in sample_annotation(cem)).

`mean_var_plot` Mean x variance scatterplot.

`hist_plot` Expression histogram.

`qq_plot` Quantile-quantile plot.

`sample_name_column` character string containing the name of the column with sample names in the annotation file.

`class_column` character string containing the name of the column with class names in the annotation file.

`mod_colors` character `vector` containing colors associated with each network module.

`parameters` `list` containing analysis parameters.

`adjacency` `matrix` containing gene adjacency values based on correlation

## Examples

```
# Get example expression data
data(expr0)
# Initialize CEMiTool object with expression
cem <- new("CEMiTool", expression=expr0)
```

cem_overlap *Integrates CEMiTool analyses*

## Description

Returns the occurrence of edges between different analyses

## Usage

```
cem_overlap(analyses, num_studies = 0, desired_table = "adjacency",
  verbose = TRUE)
```

## Arguments

| | |
|---|---|
| analyses | List of objects of class CEMiTool |
| num_studies | The minimum number of objects in the analyses list in which an edge pair must be present to be selected (default = 0) |
| desired_table | Character string indicating the type of output to be returned. Default: 'adjacency'. |
| verbose | Logical. If TRUE, reports analysis steps. |

## Details

The method assumes that all genes inside each module are connected to every other gene from the same module. Argument desired_table must be one of spearman (returns Spearman's rho), pearson (Pearson's R), b_correlations (returns adjacency list defined in CEMiTool object), adjacency (returns discretized edges)

## Value

Object of class data.frame containing edgelist describing common edges between the networks defined in module slots from the input objects

## Examples

```
## Not run:
# Run the cemitool function twice on expr dataset. Each time, one sample will be removed
data(expr0)
set.seed(10)
dset1 <- expr0[,-sample(1:ncol(expr0), 1)]
dset2 <- expr0[,-sample(1:ncol(expr0), 1)]
cem1 <- cemitool(dset1, plot=FALSE)
cem2 <- cemitool(dset2, plot=FALSE)
cem_overlap_df <- cem_overlap(list(cem1, cem2))

## End(Not run)
```

## diagnostic_report *Diagnostic report*

### Description

Creates report for identifying potential problems with data.

### Usage

```
diagnostic_report(cem, ...)

## S4 method for signature 'CEMiTool'
diagnostic_report(cem, title = "Diagnostics",
  directory = "./Reports/Diagnostics", force = FALSE, ...)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | parameters to rmarkdown::render |
| title | Character string with the title of the report. |
| directory | Directory name for results. |
| force | If the directory exists, execution will not stop. |

### Value

An HTML file with an interactive diagnostic report.

## enrich_mods *Enriches communities*

### Description

Returns enrichment of communities from edgelist created by cemoverlap function.

### Usage

```
enrich_mods(community_list, analyses, comp_group = "none",
  subject_col = NULL, run_fgsea = FALSE)
```

### Arguments

| | |
|---|---|
| community_list | Community list obtained from overlap community function |
| analyses | List of CEMiTool objects |
| comp_group | Which group will be used as base for comparison. If 'none', then all combinations of comparisons will be made |
| subject_col | Column containing subject information in sample annotation slot of CEMiTool objects |
| run_fgsea | Logical. Should fgsea be run? |

## Details

This function assumes that relevant modules for a comparison in a study will have a high proportion of differentially regulated genes in a certain direction. Base assumption is that NON-relevant modules will be centered at zero.

## Value

A `data.frame` containing information of how much each comparison is enriched in each community in each CEMiTool object.

## Examples

```
## Not run:
# Run the cemitool function twice on expr dataset. Each time, one sample will be removed
data(expr0)
data(sample_annot)
set.seed(10)
dset1 <- expr0[,-sample(1:ncol(expr0), 1)]
dset2 <- expr0[,-sample(1:ncol(expr0), 1)]
cem1 <- cemitool(dset1, plot=FALSE)
cem2 <- cemitool(dset2, plot=FALSE)
cem_overlap_df <- cem_overlap(list(cem1, cem2))

comm_overlap <- overlap_community(cem_overlap_df)

samples1 <- names(expr_data(cem1, filter=TRUE, apply_vst=TRUE))
samples2 <- names(expr_data(cem2, filter=TRUE, apply_vst=TRUE))

sample_annotation(cem1) <- sample_annot[sample_annot$SampleName %in% samples1, ]
sample_annotation(cem2) <- sample_annot[sample_annot$SampleName %in% samples2, ]

mod_enrich <- enrich_mods(comm_overlap, list(cem1, cem2), comp_group='g0')

## End(Not run)
```

---

# `expr0` *Yellow Fever gene expression data from GEO study GSE13485*

---

## Description

Modified data from a yellow fever vaccination study by Querec et al, 2009. In order to reduce package size, only the 4000 genes with the highest variance were selected for this dataset.

## Usage

```
data(expr0)
```

## Format

An object of class `data.frame`

## Source

GEO

## References

Querec TD, Akondy RS, Lee EK, Cao W et al. Systems biology approach predicts immunogenicity of the yellow fever vaccine in humans. Nat Immunol 2009 Jan;10(1):116-25. PMID: 19029902 PubMed

## Examples

```
data(expr0)
# Run CEMiTool analysis
## Not run: cemitool(expr0)
```

---

`expr_data` *Retrieve and set expression attribute*

---

## Description

Retrieve and set expression attribute

## Usage

```
expr_data(cem, ...)

## S4 method for signature 'CEMiTool'
expr_data(cem, filter = TRUE, apply_vst = FALSE,
  ...)

expr_data(cem) <- value

## S4 replacement method for signature 'CEMiTool'
expr_data(cem) <- value
```

## Arguments

| | |
|---|---|
| `cem` | Object of class `CEMiTool` |
| `...` | Additional parameters to `filter_genes` or `select_genes` functions. |
| `filter` | logical. If TRUE, retrieves filtered expression data (Default: TRUE) |
| `apply_vst` | logical. If TRUE, applies variance stabilizing transformation to expression data (Default: FALSE) |
| `value` | Object of class `data.frame` with gene expression data |

## Value

Object of class `data.frame` with gene expression data

## Examples

```
# Initialize an empty CEMiTool object
cem <- new_cem()
# Get example expression data
data(expr0)
# Add expression file to CEMiTool object
expr_data(cem) <- expr0
# Check expression file
head(expr_data(cem))
```

---

`filter_genes` *Filter a gene expression data.frame*

---

## Description

Filter a gene expression data.frame

## Usage

```
filter_genes(expr, pct = 0.75, apply_vst = FALSE)
```

## Arguments

| | |
|---|---|
| `expr` | A data.frame containing expression data |
| `pct` | Percentage of most expressed genes to keep. |
| `apply_vst` | Logical. If TRUE, will apply variance stabilizing transform before filtering data. |

## Details

This function takes a gene expression data.frame and applies a percentage filter (keeps the `pct`) most expressed genes). If `apply_vst` is TRUE, a variance stabilizing transformation is applied on gene expression values as long as mean and variance values have a Spearman's rho of over 0.5. This transformation is intended to remove this dependence between the parameters. One should then apply the `select_genes` function to get significant genes.

## Value

A data.frame containing filtered expression data

## Examples

```
# Get example expression data
data(expr0)
# Filter genes
expr_f <- filter_genes(expr0)
# Check selected genes
expr_f[1:5, 1:5]
# Filter genes and apply variance stabilizing transformation
expr_f2 <- filter_genes(expr0, apply_vst=TRUE)
# Check results
expr_f2[1:5, 1:5]
# Selected genes
```

```
selected <- select_genes(expr_f2)
# Get data.frame with only selected genes
expr_s <- expr_f2[selected, ]
# Check results
expr_s[1:5, 1:5]
```

---

`find_modules` *Co-expression modules definition*

---

## Description

Defines co-expression modules

## Usage

```
find_modules(cem, ...)

## S4 method for signature 'CEMiTool'
find_modules(cem, cor_method = c("pearson",
  "spearman"), cor_function = "cor", eps = 0.1, set_beta = NULL,
  force_beta = FALSE, min_ngen = 20, merge_similar = TRUE,
  network_type = "unsigned", tom_type = "signed", diss_thresh = 0.8,
  verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `cem` | Object of class `CEMiTool`. |
| `...` | Optional parameters. |
| `cor_method` | A character string indicating which correlation coefficient is to be computed. Default `"pearson"` |
| `cor_function` | A character string indicating the correlation function to be used. Default `'cor'`. |
| `eps` | A value for accepted R-squared interval between subsequent beta values. Default is 0.1. |
| `set_beta` | A value to override the automatically selected beta value. Default is NULL. |
| `force_beta` | Whether or not to automatically force a beta value based on number of samples. Default is FALSE. |
| `min_ngen` | Minimal number of genes per submodule. Default `20`. |
| `merge_similar` | Logical. If `TRUE`, (the default) merge similar modules. |
| `network_type` | A character string indicating to use either "unsigned" (default) or "signed" networks. Default `"unsigned"` |
| `tom_type` | A character string indicating to use either "unsigned" or "signed" (default) TOM similarity measure. |
| `diss_thresh` | Module merging correlation threshold for eigengene similarity. Default `0.8`. |
| `verbose` | Logical. If `TRUE`, reports analysis steps. Default `FALSE` |

## Value

Object of class `CEMiTool`

## Examples

```
# Get example expression data
data(expr0)
# Initialize CEMiTool object with expression
cem <- new_cem(expr0, filter=TRUE, apply_vst=FALSE)
# Define network modules
cem <- find_modules(cem)
# Check results
head(module_genes(cem))
```

---

# fit_data *Retrieve scale-free model fit data*

---

## Description

Retrieve scale-free model fit data

## Usage

```
fit_data(cem)

## S4 method for signature 'CEMiTool'
fit_data(cem)
```

## Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |

## Value

Object of class data.frame

## Examples

```
# Get example CEMiTool object
data(cem)
# Get modules and beta data
cem <- find_modules(cem)
# Get fit data
fit_data(cem)
```

## generate_report *CEMiTool report*

### Description

Creates report for CEMiTool results

### Usage

```
generate_report(cem, ...)

## S4 method for signature 'CEMiTool'
generate_report(cem, max_rows_ora = 50,
  title = "Report", directory = "./Reports/Report", force = FALSE,
  ...)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | parameters to rmarkdown::render |
| max_rows_ora | maximum number of rows in Over Representation Analysis table results |
| title | Character string with the title of the report. |
| directory | Directory name for results. |
| force | If the directory exists, execution will not stop. |

### Value

An HTML file with an interactive report of CEMiTool analyses.

### Examples

```
## Not run:
# Get example CEMiTool object
data(cem)
generate_report(cem, output_format=c("pdf_document", "html_document"))

## End(Not run)
```

## get_adj *Calculate adjacency matrix*

### Description

This function takes a CEMiTool object and returns an adjacency matrix.

### Usage

```
get_adj(cem, ...)

## S4 method for signature 'CEMiTool'
get_adj(cem, beta, network_type = "unsigned",
  cor_function = "cor", cor_method = "pearson")
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |
| ... | Optional parameters. |
| beta | Selected soft-threshold value |
| network_type | A character string indicating to use either "unsigned" (default) or "signed" networks. Default "unsigned". |
| cor_function | A character string indicating the correlation function to be used. Default 'cor'. |
| cor_method | A character string indicating which correlation coefficient is to be computed. Default "pearson". |

### Value

Object of class CEMiTool with adjacency data

### Examples

```
# Get example expression data
data(expr0)
# Initialize new CEMiTool object with expression data
cem <- new_cem(expr0, filter=TRUE, apply_vst=FALSE)
# Calculate adjacency matrix with example beta value 8
cem <- get_adj(cem, beta=8)
# Check results
adj <- adj_data(cem)
adj[1:5, 1:5]
```

---

## get_beta_data *Soft-threshold beta data*

---

### Description

This function takes the input parameters from find_modules and calculates the WGCNA soft-threshold parameters and returns them.

### Usage

```
get_beta_data(cem, ...)

## S4 method for signature 'CEMiTool'
get_beta_data(cem, network_type = "unsigned",
  cor_function = "cor", cor_method = "pearson", verbose = FALSE)
```

### Arguments

| | |
|---|---|
| cem | A CEMiTool object containing expression data |
| ... | Optional parameters. |
| network_type | A character string indicating to use either "unsigned" (default) or "signed" networks. Default "unsigned". |
| cor_function | A character string indicating the correlation function to be used. Default 'cor'. |
| cor_method | A character string indicating which correlation coefficient is to be computed. Default "pearson" |
| verbose | Logical. If TRUE, reports analysis steps. Default FALSE |

### Value

A list containing the soft-threshold selected by WGCNA and scale-free model parameters

### Examples

```
# Get example expression data
data(expr0)
# Initialize new CEMiTool object with expression data
cem <- new_cem(expr0, filter=TRUE, apply_vst=FALSE)
# Get beta data
beta_data <- get_beta_data(cem)
```

---

## get_cemitool_r2_beta *Calculate CEMiTool beta and R2 values*

---

### Description

This function takes a CEMiTool object with beta data and returns a vector containing the chosen beta and corresponding R squared value.

### Usage

```
get_cemitool_r2_beta(cem, ...)

## S4 method for signature 'CEMiTool'
get_cemitool_r2_beta(cem, eps = 0.1)
```

### Arguments

| | |
|---|---|
| cem | A CEMiTool object containing the fit_indices slot |
| ... | Optional parameters. |
| eps | A value indicating the accepted interval between successive values of R squared to use to calculate the selected beta. Default: 0.1. |

### Value

A vector containing R squared value and the chosen beta parameter.

## Examples

```
# Get example expression data
data(expr0)
# Initialize new CEMiTool object with expression data
cem <- new_cem(expr0, filter=TRUE, apply_vst=FALSE)
# Get modules and beta data
cem <- find_modules(cem)
# Get CEMiTool R2 and beta values
get_cemitool_r2_beta(cem)
```

---

# get_connectivity *Calculate network connectivity*

---

## Description

This function takes a CEMiTool object and returns the network connectivity.

## Usage

```
get_connectivity(cem, ...)

## S4 method for signature 'CEMiTool'
get_connectivity(cem, beta)
```

## Arguments

| | |
|---|---|
| cem | Object of class CEMiTool containing the fit_indices slot |
| ... | Optional parameters. |
| beta | A soft-thresholding value to be used for the network. |

## Value

The value of the network's connectivity.

## Examples

```
# Get example expression data
data(expr0)
# Initialize new CEMiTool object with expression data
cem <- new_cem(expr0, filter=TRUE, apply_vst=FALSE)
# Get modules and beta data
cem <- find_modules(cem)
# Get network connectivity with example beta value 8
get_connectivity(cem, beta=8)
```

## get_hubs *Get hubs*

### Description

Returns n genes in each module with high connectivity.

### Usage

```
get_hubs(cem, ...)

## S4 method for signature 'CEMiTool'
get_hubs(cem, n = 5, method = "adjacency")
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | Optional parameters. |
| n | Number of hubs to return from each module. If "all", returns all genes in decreasing order of connectivity. Default: 5. |
| method | Method for hub calculation. Either "adjacency" or "kME". Default: "adjacency" |

### Value

A list containing hub genes for each module and the value of the calculated method.

### Examples

```
# Get example CEMiTool object
data(cem)
# Get module hubs
hubs <- get_hubs(cem, n=10, "adjacency")
```

## get_merged_mods *Merge similar modules*

### Description

This function takes a CEMiTool object with expression and a vector of numeric labels to merge similar modules.

### Usage

```
get_merged_mods(cem, ...)

## S4 method for signature 'CEMiTool'
get_merged_mods(cem, mods, diss_thresh = 0.8,
  verbose = FALSE)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | Optional parameters. |
| mods | A vector containing numeric labels for each module gene |
| diss_thresh | A threshold of dissimilarity for modules. Default is 0.8. |
| verbose | Logical. If TRUE, reports analysis steps. Default FALSE |

### Value

Numeric labels assigning genes to modules.

### Examples

```
# Get example expression data
data(expr0)
# Initialize new CEMiTool object with expression data
cem <- new_cem(expr0, filter=TRUE, apply_vst=FALSE)
# Calculate adjacency matrix with example beta value 8
cem <- get_adj(cem, beta=8)
# Get modules
mods <- get_mods(cem)
# Merge similar modules
merged_mods <- get_merged_mods(cem, mods)
```

---

## get_mods *Calculate co-expression modules*

---

### Description

This function takes a CEMiTool object containing an adjacency matrix together with the given network parameters, and returns the given co-expression modules.

### Usage

```
get_mods(cem, ...)

## S4 method for signature 'CEMiTool'
get_mods(cem, cor_function = "cor",
  cor_method = "pearson", tom_type = "signed", min_ngen = 20)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | Optional parameters. |
| cor_function | A character string indicating the correlation function to be used. Default 'cor'. |
| cor_method | A character string indicating which correlation coefficient is to be computed. Default "pearson". |
| tom_type | A character string indicating to use either "unsigned" or "signed" (default) TOM similarity measure. |
| min_ngen | Minimal number of genes per module (Default: 20). |

## Value

Numeric labels assigning genes to modules.

## Examples

```
# Get example expression data
data(expr0)
# Initialize new CEMiTool object with expression data
cem <- new_cem(expr0, filter=TRUE, apply_vst=FALSE)
# Calculate adjacency matrix with example beta value 8
cem <- get_adj(cem, beta=8)
# Get module labels
mods <- get_mods(cem)
```

---

`get_phi` *Calculate phi*

---

## Description

This function takes a CEMiTool object and returns the phi parameter.

## Usage

```
get_phi(cem, ...)

## S4 method for signature 'CEMiTool'
get_phi(cem)
```

## Arguments

| | |
|---|---|
| `cem` | A CEMiTool object containing the fit_indices slot |
| `...` | Optional parameters. |

## Value

The phi parameter

## Examples

```
# Get example expression data
data(expr0)
# Initialize new CEMiTool object with expression data
cem <- new_cem(expr0, filter=TRUE, apply_vst=FALSE)
# Get modules and beta data
cem <- find_modules(cem)
# Get phi
get_phi(cem)
```

## gsea_data *Retrieve Gene Set Enrichment Analysis (GSEA) results*

### Description

Retrieve Gene Set Enrichment Analysis (GSEA) results

### Usage

```
gsea_data(cem)

## S4 method for signature 'CEMiTool'
gsea_data(cem)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |

### Value

Object of class list with GSEA data

### Examples

```
# Get example CEMiTool object
data(cem)
# Look at example annotation file
sample_annotation(cem)
# Run GSEA on network modules
cem <- mod_gsea(cem)
# Check results
gsea_data(cem)
```

## interactions_data *Retrieve and set interaction data to CEMiTool object*

### Description

Retrieve and set interaction data to CEMiTool object

### Usage

```
interactions_data(cem, ...)

## S4 method for signature 'CEMiTool'
interactions_data(cem)

interactions_data(cem) <- value

## S4 replacement method for signature 'CEMiTool'
interactions_data(cem) <- value
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | parameters for igraph::graph_from_data_frame |
| value | a data.frame or matrix containing two columns |

### Value

Object of class CEMiTool

### Examples

```
# Get example CEMiTool object
data(cem)
# Read example interactions data
int_df <- read.delim(system.file("extdata", "interactions.tsv",
    package = "CEMiTool"))
# Insert interactions data
interactions_data(cem) <- int_df
# Check interactions data
interactions_data(cem)
```

---

## module_genes *Get the module genes in a CEMiTool object*

---

### Description

Get the module genes in a CEMiTool object

### Usage

```
module_genes(cem, module = NULL)

## S4 method for signature 'CEMiTool'
module_genes(cem, module = NULL)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |
| module | A character string with the name of the module of which genes are to be returned. Defaults to NULL, which returns the full list of genes and modules. |

### Value

Object of class data.frame containing genes and their respective module

## Examples

```
# Get example CEMiTool object
data(cem)
# Get the module genes
module_genes(cem)
# Get genes for module M1
module_genes(cem, module="M1")
```

---

mod_colors *Retrieve and set mod_colors attribute*

---

## Description

Retrieve and set mod_colors attribute

## Usage

```
mod_colors(cem)

## S4 method for signature 'CEMiTool'
mod_colors(cem)

mod_colors(cem) <- value

## S4 replacement method for signature 'CEMiTool,character'
mod_colors(cem) <- value
```

## Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |
| value | a character vector containing colors for each module. Names should match with module names |

## Value

A vector with color names.

## Examples

```
# Get example CEMiTool object
data(cem)
# See module colors
mod_colors(cem)
```

## mod_gene_num *Get the number of genes in modules in a CEMiTool object*

### Description

Get the number of genes in modules in a CEMiTool object

### Usage

```
mod_gene_num(cem, module = NULL)

## S4 method for signature 'CEMiTool'
mod_gene_num(cem, module = NULL)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |
| module | Default is NULL. If a character string designating a module is given, the number of genes in that module is returned instead. |

### Value

The number of genes in module(s)

### Examples

```
# Get example CEMiTool object
data(cem)
# Get the number of genes in modules
mod_gene_num(cem)
# Get the number of genes in module M1
mod_gene_num(cem, "M1")
```

## mod_gsea *Module Gene Set Enrichment Analysis*

### Description

Perfoms Gene Set Enrichment Analysis (GSEA) for each co-expression module found.

### Usage

```
mod_gsea(cem, ...)

## S4 method for signature 'CEMiTool'
mod_gsea(cem, gsea_scale = TRUE,
  rank_method = "mean", gsea_min_size = 15, gsea_max_size = 1000,
  verbose = FALSE)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | Optional parameters. |
| gsea_scale | If TRUE, transform data using z-score transformation. Default: TRUE |
| rank_method | Character string indicating how to rank genes. Either "mean" (the default) or "median". |
| gsea_min_size | Minimum gene set size (Default: 15). |
| gsea_max_size | Maximum gene set size (Default: 1000). |
| verbose | logical. Report analysis steps. |

### Value

GSEA results.

### See Also

plot_gsea

### Examples

```
# Get example CEMiTool object
data(cem)
# Look at example annotation file
sample_annotation(cem)
# Run GSEA on network modules
cem <- mod_gsea(cem)
# Check results
gsea_data(cem)
```

---

## mod_names *Get module names in a CEMiTool object*

---

### Description

Get module names in a CEMiTool object

### Usage

```
mod_names(cem, include_NC = TRUE)

## S4 method for signature 'CEMiTool'
mod_names(cem, include_NC = TRUE)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |
| include_NC | Logical. Whether or not to include "Not.Correlated" module. Defaults to TRUE. |

## Value

Module names

## Examples

```
# Get example CEMiTool object
data(cem)
# Get module names
mod_names(cem)
```

---

`mod_ora` *Module Overrepresentation Analysis*

---

## Description

Performs overrepresentation analysis for each co-expression module found.

## Usage

```
mod_ora(cem, ...)

## S4 method for signature 'CEMiTool'
mod_ora(cem, gmt, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `cem` | Object of class `CEMiTool`. |
| `...` | Optional parameters. |
| `gmt` | Object of class `data.frame` with 2 columns, one with pathways and one with genes |
| `verbose` | logical. Report analysis steps. |

## Value

Object of class `CEMiTool`

## See Also

`ora_data`

## Examples

```
# Get example CEMiTool object
data(cem)
# Read gmt file
gmt <- read_gmt(system.file('extdata', 'pathways.gmt',
                  package='CEMiTool'))
# Run module overrepresentation analysis
cem <- mod_ora(cem, gmt)
# Check results
head(ora_data(cem))
```

mod_summary *Co-expression module summarization*

## Description

Summarizes modules using mean or eigengene expression.

## Usage

```
mod_summary(cem, ...)

## S4 method for signature 'CEMiTool'
mod_summary(cem, method = c("mean", "median",
  "eigengene"), verbose = FALSE)
```

## Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | Optional parameters. |
| method | A character string indicating which summarization method is to be used. Can be 'eigengene', 'mean' or 'median'. Default is 'mean'. |
| verbose | Logical. If TRUE, reports analysis steps. |

## Value

A data.frame with summarized values.

## Examples

```
# Get example CEMiTool object
data(cem)
# Summarize results
mod_summary <- mod_summary(cem)
```

new_cem *Create a CEMiTool object*

## Description

Create a CEMiTool object

## Usage

```
new_cem(expr = data.frame(), sample_annot = data.frame(),
  sample_name_column = "SampleName", class_column = "Class",
  filter = TRUE, apply_vst = FALSE)
```

### Arguments

| | |
|---|---|
| expr | Object of class data.frame with gene expression data |
| sample_annot | Object of data.frame containing the sample annotation. It should have at least two columns containing group Class and the Sample Name that should match with samples in expression file |
| sample_name_column | A string specifying the column to be used as sample identification. Default: "SampleName". |
| class_column | A string specifying the column to be used as a grouping factor for samples. Default: "Class" |
| filter | Logical. Used to define if posterior functions should use filtered expression data or not (Default: TRUE) |
| apply_vst | Logical. Used to define if posterior functions should use a variance stabilizing transformation on expression data before analyses. Only valid if argument filter is TRUE. (Default: FALSE) |

### Value

Object of class CEMiTool

### Examples

```
# Create new CEMiTool object
cem <- new_cem()
# Create new CEMiTool object with expression and sample_annotation data
data(expr0)
data(sample_annot)
cem <- new_cem(expr0, sample_annot, "SampleName", "Class")
# Equivalent to a call to new()
cem2 <- new("CEMiTool", expression=expr0, sample_annotation=sample_annot)
identical(cem, cem2)
```

---

## nmodules *Get the number of modules in a CEMiTool object*

---

### Description

Get the number of modules in a CEMiTool object

### Usage

```
nmodules(cem)

## S4 method for signature 'CEMiTool'
nmodules(cem)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |

## Value

number of modules

## Examples

```
# Get example CEMiTool object
data(cem)
# Get the number of modules
nmodules(cem)
```

---

`ora_data` *Retrieve over representation analysis (ORA) results*

---

## Description

Retrieve over representation analysis (ORA) results

## Usage

```
ora_data(cem)

## S4 method for signature 'CEMiTool'
ora_data(cem)
```

## Arguments

| | |
|---|---|
| `cem` | Object of class `CEMiTool` |

## Details

This function returns the results of the `mod_ora` function on the `CEMiTool` object. The ID column corresponds to pathways in the gmt file for which genes in the modules were enriched. The Count column shows the number of genes in the module that are enriched for each pathway. The GeneRatio column shows the proportion of genes in the module enriched for a given pathway out of all the genes in the module enriched for any given pathway. The BgRatio column shows the proportion of genes in a given pathway out of all the genes in the gmt file. For more details, please refer to the `clusterProfiler` package documentation.

## Value

Object of class `data.frame` with ORA data

## References

Guangchuang Yu, Li-Gen Wang, Yanyan Han, Qing-Yu He. clusterProfiler: an R package for comparing biological themes among gene clusters. OMICS: A Journal of Integrative Biology. 2012, 16(5):284-287.

## Examples

```
# Get example CEMiTool object
data(cem)
# Read gmt file
gmt <- read_gmt(system.file('extdata', 'pathways.gmt',
                  package='CEMiTool'))
# Run module overrepresentation analysis
cem <- mod_ora(cem, gmt)
# Check results
head(ora_data(cem))
```

---

`overlap_community` *Generates communities from edgelist*

---

## Description

Returns communities from edgelist created by cemoverlap function.

## Usage

```
overlap_community(mod_intersection_df, presence_as_weights = FALSE,
  smallest_community = 15, method = c("cluster_fast_greedy",
  "cluster_edge_betweenness", "cluster_label_prop",
  "cluster_leading_eigen", "cluster_louvain", "cluster_optimal",
  "cluster_spinglass", "cluster_walktrap"))
```

## Arguments

| | |
|---|---|
| `mod_intersection_df` | Module intersection dataframe obtained from cemoverlap function |
| `presence_as_weights` | Logical. Should sums of node pair occurence be considered edge weights? |
| `smallest_community` | Minimal number of genes in community (default:15) |
| `method` | Character string denoting an `igraph` package clustering function. One of 'cluster_fast_greedy', 'cluster_edge_betweenness', 'cluster_fast_greedy', 'cluster_label_prop', 'cluster_leading_eigen', 'cluster_louvain', 'cluster_optimal', 'cluster_spinglass' or 'cluster_walktrap'. Default: 'cluster_fast_greedy' |

## Details

Function takes edgelist as inputs and generates communities using functions provided in igraph package (default:'cluster_fast_greedy')

## Value

A list containing the genes present in each community detected

## Examples

```
## Not run:
# Run the cemitool function twice on expr dataset. Each time, one sample will be removed
data(expr0)
set.seed(10)
dset1 <- expr0[,-sample(1:ncol(expr0), 1)]
dset2 <- expr0[,-sample(1:ncol(expr0), 1)]
cem1 <- cemitool(dset1, plot=FALSE)
cem2 <- cemitool(dset2, plot=FALSE)
cem_overlap_df <- cem_overlap(list(cem1, cem2))
comm_overlap_df <- overlap_community(cem_overlap_df)

## End(Not run)
```

---

# plot_beta_r2 *Soft-threshold beta selection graph*

---

## Description

Creates a graph showing each possible soft-threshold value and its corresponding R squared value

## Usage

```
plot_beta_r2(cem, ...)

## S4 method for signature 'CEMiTool'
plot_beta_r2(cem,
  plot_title = "Scale independence (beta selection)")
```

## Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | Optional parameters. |
| plot_title | title of the graph |

## Value

Object of class CEMiTool with beta x R squared plot

## Examples

```
# Get example CEMiTool object
data(cem)
# Plot scale-free model fit as a function of the soft-thresholding beta parameter choice
cem <- plot_beta_r2(cem)
# Check resulting plot
show_plot(cem, "beta_r2")
```

plot_comembership *Plot module edge co-membership*

## Description

Plot module edge co-membership

## Usage

```
plot_comembership(cem_overlap_df)
```

## Arguments

| | |
|---|---|
| cem_overlap_df | Output of function cem_overlap |

## Value

A plot containing the number of edges present in each fraction of studies

## Examples

```
## Not run:
# Run the cemitool function five times on expr0 dataset. Each time, 10 samples will be removed.
data(expr0)
data(sample_annot)
set.seed(10)
dset1 <- expr0[,-sample(1:ncol(expr0), 10)]
set.seed(11)
dset2 <- expr0[,-sample(1:ncol(expr0), 10)]
set.seed(12)
dset3 <- expr0[,-sample(1:ncol(expr0), 10)]
set.seed(13)
dset4 <- expr0[,-sample(1:ncol(expr0), 10)]
set.seed(14)
dset5 <- expr0[,-sample(1:ncol(expr0), 10)]

cem1 <- cemitool(dset1, sample_annot, plot=FALSE)
cem2 <- cemitool(dset2, sample_annot, plot=FALSE)
cem3 <- cemitool(dset3, sample_annot, plot=FALSE)
cem4 <- cemitool(dset4, sample_annot, plot=FALSE)
cem5 <- cemitool(dset5, sample_annot, plot=FALSE)

cem_overlap_df <- cem_overlap(list(cem1, cem2, cem3, cem4, cem5))
plot_comembership(cem_overlap_df)

## End(Not run)
```

plot_consensus *Plot graph of consensus modules*

## Description

Plot graph of consensus modules

## Usage

```
plot_consensus(cem_overlap_df, comm_overlap_df, study_num,
  num_sd_cut = 2)
```

## Arguments

| | |
|---|---|
| cem_overlap_df | Output data.frame from function cem_overlap |
| comm_overlap_df | Output data.frame from function overlap_community |
| study_num | Minimum number of studies an edge must be present in for it to be included |
| num_sd_cut | Number of standard deviations an edge's mean must be above in order to be included in the final plot (Default: 2) |

## Value

A plot containing the consensus modules of the studies

## Examples

```
## Not run:
# Run the cemitool function twice on expr dataset. Each time, one sample will be removed
data(expr0)
data(sample_annot)
set.seed(10)
dset1 <- expr0[,-sample(1:ncol(expr0), 1)]
dset2 <- expr0[,-sample(1:ncol(expr0), 1)]
cem1 <- cemitool(dset1, sample_annot, plot=FALSE)
cem2 <- cemitool(dset2, sample_annot, plot=FALSE)
cem_overlap_df <- cem_overlap(list(cem1, cem2))
comm_overlap_df <- overlap_community(cem_overlap_df)
plot_consensus(cem_overlap_df, comm_overlap_df, study_num=2)

## End(Not run)
```

## plot_gsea *GSEA visualization*

### Description

Creates a heatmap with the results of gene set enrichment analysis (GSEA) of co-expression modules

### Usage

```
plot_gsea(cem, ...)

## S4 method for signature 'CEMiTool'
plot_gsea(cem, pv_cut = 0.05)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | Optional parameters. |
| pv_cut | P-value cut-off. Default 0.05 |

### Value

Object of class CEMiTool with GSEA plots

### Examples

```
# Get example CEMiTool object
data(cem)
# Get example sample annotation file
# Run GSEA on network modules
cem <- mod_gsea(cem)
# Plot GSEA results
cem <- plot_gsea(cem)
# Check resulting plot
show_plot(cem, "gsea")
```

## plot_hist *Plot histogram*

### Description

This function plots a histogram of the distribution of gene expression, to help assess the normality of the data.

### Usage

```
plot_hist(cem, ...)

## S4 method for signature 'CEMiTool'
plot_hist(cem, filter = FALSE)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |
| ... | Optional parameters |
| filter | Logical. Whether or not to use filtered data for CEMiTool objects (Default: FALSE). |

### Value

Object of class CEMiTool containing expression histogram

### Examples

```
# Get example CEMiTool object
data(cem)
# Plot histogram
cem <- plot_hist(cem)
# Check results
show_plot(cem, "hist")
```

---

## plot_interactions *Network visualization*

---

### Description

Creates a graph based on interactions provided

### Usage

```
plot_interactions(cem, ...)

## S4 method for signature 'CEMiTool'
plot_interactions(cem, n = 10, ...)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | Optional parameters. |
| n | number of nodes to label |

### Value

Object of class CEMiTool with profile plots

## Examples

```
# Get example CEMiTool object
data(cem)
# Get example gene interactions data
int <- system.file("extdata", "interactions.tsv", package = "CEMiTool")
int_df <- read.delim(int)
# Include interaction data into CEMiTool object
interactions_data(cem) <- int_df
# Plot resulting networks
cem <- plot_interactions(cem)
# Check resulting plot
show_plot(cem, "interaction")
```

---

# plot_mean_k *Network mean connectivity*

---

## Description

Creates a graph showing the mean connectivity of genes in the network

## Usage

```
plot_mean_k(cem, ...)

## S4 method for signature 'CEMiTool'
plot_mean_k(cem, title = "Mean connectivity")
```

## Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | Optional parameters. |
| title | title of the graph |

## Value

Object of class CEMiTool with connectivity plot

## Examples

```
# Get example CEMiTool object
data(cem)
# Plot scale-free model fit as a function of the soft-thresholding beta parameter choice
cem <- plot_mean_k(cem)
# Check resulting plot
show_plot(cem, "mean_k")
```

## plot_mean_var *Plot mean and variance*

### Description

This plot returns a scatterplot of the mean by the variance of gene expression. A linear relationship between these values for RNAseq data suggest that an appropriate transformation such as the Variance Stabilizing Transformation should be applied.

### Usage

```
plot_mean_var(cem, ...)

## S4 method for signature 'CEMiTool'
plot_mean_var(cem, filter = FALSE)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |
| ... | Optional parameters |
| filter | Logical. Whether or not to use filtered data for CEMiTool objects (Default: FALSE). |

### Value

Object of class CEMiTool containing a mean and variance plot

### Examples

```
# Get example CEMiTool object
data(cem)
# Plot mean and variance plot
cem <- plot_mean_var(cem)
# Check results
show_plot(cem, 'mean_var')
```

## plot_ora *ORA visualization*

### Description

Creates a bar plot with the results of module overrepresentation analysis

### Usage

```
plot_ora(cem, ...)

## S4 method for signature 'CEMiTool'
plot_ora(cem, n = 10, pv_cut = 0.05, ...)
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | parameters to plot_ora_single |
| n | number of modules to show |
| pv_cut | p-value significance cutoff. Default is 0.05. |

### Value

Object of class CEMiTool with ORA plots

### Examples

```
# Get example CEMiTool object
data(cem)
# Read example gmt file
gmt <- read_gmt(system.file('extdata', 'pathways.gmt',
                  package='CEMiTool'))
# Run overrepresentation analysis
cem <- mod_ora(cem, gmt)
# Plot module gene expression profiles
cem <- plot_ora(cem)
# Check resulting plot
show_plot(cem, "ora")
```

---

## plot_profile *Expression profile visualization*

---

### Description

Creates a plot with module gene expression profiles along samples

### Usage

```
plot_profile(cem, ...)

## S4 method for signature 'CEMiTool'
plot_profile(cem, order_by_class = TRUE,
  center_func = "mean")
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| ... | Optional parameters. |
| order_by_class | Logical. Only used if a sample annotation file is present. Whether or not to order by the class column in the sample annotation file (as defined by the class_column slot in cem). |
| center_func | Character string indicating the centrality measure to show in the plot. Either 'mean' (the default) or 'median'. |

## Value

Object of class CEMiTool with profile plots

## Examples

```
# Get example CEMiTool object
data(cem)
# Plot module gene expression profiles
cem <- plot_profile(cem)
# Check resulting plot
show_plot(cem, "profile")
```

---

plot_qq *Plot quantile-quantile plot*

---

## Description

This function creates a normal QQ plot of the expression values.

## Usage

```
plot_qq(cem, ...)

## S4 method for signature 'CEMiTool'
plot_qq(cem, filter = FALSE)
```

## Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |
| ... | Optional parameters |
| filter | Logical. Whether or not to use filtered data for CEMiTool objects (Default: FALSE). |

## Value

Object of class CEMiTool containing qqplot

## Examples

```
# Get example CEMiTool object
data(cem)
# Plot quantile-quantile plot
cem <- plot_qq(cem)
# Check results
show_plot(cem, 'qq')
```

plot_sample_tree *Sample clustering*

## Description

Creates a dendrogram showing the similarities between samples in the expression data.

## Usage

```
plot_sample_tree(cem, ...)

## S4 method for signature 'CEMiTool'
plot_sample_tree(cem, col_vector = NULL,
  sample_name_column = NULL, class_column = NULL, filter = FALSE)
```

## Arguments

| | |
|---|---|
| cem | Object of class CEMiTool or data.frame. |
| ... | Optional parameters. |
| col_vector | A vector of columns to use for visualizing the clustering. See Details. |
| sample_name_column | A string specifying the column to be used as sample identification. For CEMiTool objects, this will be the string specified in the sample_name_column slot. |
| class_column | A string specifying the column to be used as sample group identification. For CEMiTool objects, this will be the string specified in the class_column slot. |
| filter | Logical. Whether or not to use filtered data for CEMiTool objects (Default: FALSE). |

## Value

Object of class CEMiTool with dendrogram or a plot object.

## Examples

```
# Get example CEMiTool object
data(cem)
# Plot sample dendrogram
cem <- plot_sample_tree(cem)
# Check resulting plot
show_plot(cem, "sample_tree")
```

`plot_similarity` *Plot study module similarity*

## Description

Plot study module similarity

## Usage

```
plot_similarity(mod_stats, weight_col = "logfdr")
```

## Arguments

| | |
|---|---|
| `mod_stats` | List output from function `stat_overlap_mods` |
| `weight_col` | Character string denoting the weighting column for module similarity. One of "Jaccard", "Fisherp", "fdr", "logp" or "logfdr". Default: "logfdr". |

## Value

A plot showing the similarity between study modules

## Examples

```
## Not run:
# Run the cemitool function five times on expr0 dataset. Each time, 10 samples will be removed.
data(expr0)
data(sample_annot)
set.seed(10)
dset1 <- expr0[,-sample(1:ncol(expr0), 10)]
set.seed(11)
dset2 <- expr0[,-sample(1:ncol(expr0), 10)]
set.seed(12)
dset3 <- expr0[,-sample(1:ncol(expr0), 10)]
set.seed(13)
dset4 <- expr0[,-sample(1:ncol(expr0), 10)]
set.seed(14)
dset5 <- expr0[,-sample(1:ncol(expr0), 10)]

cem1 <- cemitool(dset1, sample_annot, plot=FALSE)
cem2 <- cemitool(dset2, sample_annot, plot=FALSE)
cem3 <- cemitool(dset3, sample_annot, plot=FALSE)
cem4 <- cemitool(dset4, sample_annot, plot=FALSE)
cem5 <- cemitool(dset5, sample_annot, plot=FALSE)
mod_stats <- stat_overlap_mods(list(cem1, cem2, cem3, cem4, cem5), comp_group="g0")
plot_similarity(mod_stats)

## End(Not run)
```

## read_gmt *Read a GMT file*

### Description

Read a GMT file

### Usage

```
read_gmt(fname)
```

### Arguments

| | |
|---|---|
| fname | GMT file name. |

### Value

A list containing genes and description of each pathway

### Examples

```
# Read example gmt file
gmt_fname <- system.file("extdata", "pathways.gmt", package = "CEMiTool")
gmt_in <- read_gmt(gmt_fname)
```

## sample_annot *Yellow Fever Sample Annotation data*

### Description

Modified data from a yellow fever vaccination study by Querec et al, 2009. This dataset, together with `expr` can be used as input for CEMiTool functions

### Usage

```
data(sample_annot)
```

### Format

An object of class `data.frame`

### Source

GEO

### References

Querec TD, Akondy RS, Lee EK, Cao W et al. Systems biology approach predicts immunogenicity of the yellow fever vaccine in humans. Nat Immunol 2009 Jan;10(1):116-25. PMID: 19029902 PubMed

## Examples

```
data(expr)
data(sample_annot)
# Run CEMiTool analysis
## Not run: cemitool(expr, sample_annot)
```

---

`sample_annotation` *Retrive or set the sample_annotation attribute*

---

## Description

Retrive or set the sample_annotation attribute

## Usage

```
sample_annotation(cem)

## S4 method for signature 'CEMiTool'
sample_annotation(cem)

sample_annotation(cem, sample_name_column = "SampleName",
  class_column = "Class") <- value

## S4 replacement method for signature 'CEMiTool'
sample_annotation(cem,
  sample_name_column = "SampleName", class_column = "Class") <- value
```

## Arguments

| | |
|---|---|
| `cem` | Object of class `CEMiTool` |
| `sample_name_column` | A string containing the name of a column which should be used as a unique identifier for samples in the file. Only used when assigning a sample annotation data.frame. Default: "SampleName". |
| `class_column` | A string containing the name of a column which should be used to identify different sample groups. Only used when assigning a sample annotation data.frame. Default: "Class" |
| `value` | A data.frame containing the sample annotation, should have at least two columns containing the Class and the Sample Name that should match with samples in expression |

## Value

A data.frame containing characteristics of each sample.

## Examples

```
# Get example expression data
data(expr0)
# Get example sample_annotation data
data(sample_annot)
# Initialize CEMiTool object with expression
cem <- new_cem(expr0)
# Add sample annotation file to CEMiTool object
sample_annotation(cem,
    sample_name_column="SampleName",
    class_column="Class") <- sample_annot
# Check annotation
head(sample_annotation(cem))
```

---

`save_plots` *Save CEMiTool object plots*

---

## Description

Save plots into the directory specified by the `directory` argument.

## Usage

```
save_plots(cem, ...)

## S4 method for signature 'CEMiTool'
save_plots(cem, value = c("all", "profile", "gsea",
  "ora", "interaction", "beta_r2", "mean_k", "sample_tree", "mean_var",
  "hist", "qq"), force = FALSE, directory = "./Plots")
```

## Arguments

| | |
|---|---|
| `cem` | Object of class `CEMiTool`. |
| `...` | Optional parameters One of "all", "profile", "gsea", "ora", "interaction", "beta_r2", "mean_k", "sample_tree", "mean_var", "hist", "qq". |
| `value` | A character string containing the name of the plot to be saved. |
| `force` | If the directory exists, execution will not stop. |
| `directory` | Directory into which the files will be saved. |

## Value

A pdf file or files with the desired plot(s)

## Examples

```
# Get example CEMiTool object
data(cem)
# Plot beta x R squared graph
cem <- plot_beta_r2(cem)
# Save plot
## Not run: save_plots(cem, value="beta_r2", directory="./Plots")
```

## select_genes *Select genes based on variance*

### Description

Select genes based on variance

### Usage

```
select_genes(expr, n_genes, filter_pval = 0.1)
```

### Arguments

| | |
|---|---|
| expr | A data.frame containing expression values |
| n_genes | (Optional) Number of genes to be selected |
| filter_pval | P-value cutoff for gene selection |

### Value

A vector containing the names of selected genes

### Examples

```
# Get example expression data
data(expr0)
# Filter genes
expr_f <- filter_genes(expr0)
# Check selected genes
expr_f[1:5, 1:5]
# Filter genes and apply variance stabilizing transformation
expr_f2 <- filter_genes(expr0, apply_vst=TRUE)
# Check results
expr_f2[1:5, 1:5]
# Selected genes
selected <- select_genes(expr_f2)
# Get data.frame with only selected genes
expr_s <- expr_f2[selected, ]
# Check results
expr_s[1:5, 1:5]
```

## show,CEMiTool-method *Print a cemitool object*

### Description

Print a cemitool object

### Usage

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

### Arguments

| | |
|---|---|
| object | Object of class CEMiTool |

### Value

A CEMiTool object.

---

## show_plot *Retrieve CEMiTool object plots*

---

### Description

Retrieve CEMiTool object plots

### Usage

```
show_plot(cem, value)

## S4 method for signature 'CEMiTool'
show_plot(cem, value = c("profile", "gsea", "ora",
  "interaction", "beta_r2", "mean_k", "sample_tree", "mean_var", "hist",
  "qq"))
```

### Arguments

| | |
|---|---|
| cem | Object of class CEMiTool. |
| value | A character string containing the name of the plot to be shown. One of "profile", "gsea", "ora", "interaction", "beta_r2", "mean_k", "sample_tree", "mean_var", "hist", "qq". |

### Value

A plot corresponding to a CEMiTool analysis

### Examples

```
# Get example CEMiTool object
data(cem)
# Plot beta x R squared graph
cem <- plot_beta_r2(cem)
# Check plot
show_plot(cem, "beta_r2")
```

## stat_overlap_mods *Calculate module overlap statistics*

### Description

Calculate module overlap statistics

### Usage

```
stat_overlap_mods(analyses, comp_group, subject_col = NULL, ...)
```

### Arguments

| | |
|---|---|
| analyses | List of CEMiTool objects |
| comp_group | Character string indicating the group to be compared against |
| subject_col | Optional character string indicating a column in the CEMiTool objects' sample annotation slot object containing subject information |
| ... | Additional parameters |

### Value

A list containing overlap statistics, node and study information and module activity

### Examples

```
## Not run:
# Run the cemitool function twice on expr dataset. Each time, one sample will be removed
data(expr0)
set.seed(10)
dset1 <- expr0[,-sample(1:ncol(expr0), 1)]
dset2 <- expr0[,-sample(1:ncol(expr0), 1)]
cem1 <- cemitool(dset1, sample_annot, plot=FALSE)
cem2 <- cemitool(dset2, sample_annot, plot=FALSE)
mod_stats <- stat_overlap_mods(analyses=list(cem1, cem2), comp_group="g0")

## End(Not run)
```

## write_files *Save the CEMiTool object in files*

### Description

Save the CEMiTool object in files

### Usage

```
write_files(cem, ...)

## S4 method for signature 'CEMiTool'
write_files(cem, directory = "./Tables",
  force = FALSE)
```

## Arguments

| | |
|---|---|
| cem | Object of class CEMiTool |
| ... | Optional parameters |
| directory | a directory |
| force | if the directory exists the execution will not stop |

## Value

A directory containing CEMiTool results in files.

## Examples

```
# Get example CEMiTool object
data(cem)
# Save CEMiTool results in files
write_files(cem, directory=".", force=TRUE)
```

# Index