# Package 'tmod'

May 8, 2026

**Type** Package

**Title** Feature Set Enrichment Analysis for Metabolomics and Transcriptomics

**Version** 0.50.13

**Maintainer** January Weiner <january.weiner@gmail.com>

**Description** Methods and feature set definitions for feature or gene set enrichment analysis in transcriptional and metabolic profiling data. Package includes tests for enrichment based on ranked lists of features, functions for visualisation and multivariate functional analysis. See Zyla et al (2019) <doi:10.1093/bioinformatics/btz447>.

**URL** https://tmod.online, https://github.com/january3/tmod/, https://january3.github.io/tmod/

**License** GPL (>= 2.0)

**Depends** R (>= 2.10)

**LazyData** false

**VignetteBuilder** knitr

**Imports** beeswarm,tagcloud,XML,methods,plotwidgets,RColorBrewer,gplots,tibble,pheatmap,ggplot2,tidyr,purrr,rlang,tidyselect,g

**Suggests** testthat,knitr,rmarkdown,dplyr,pander,cowplot

**RoxygenNote** 7.2.0

**Encoding** UTF-8

**NeedsCompilation** no

**Author** January Weiner [aut, cre] (ORCID: <https://orcid.org/0000-0003-1438-7819>)

**Repository** CRAN

**Date/Publication** 2023-03-31 12:50:02 UTC

# Contents

## tmod-package *Transcriptional Module Analysis*

### Description

Transcriptional Module Analysis

### Details

The primary role of this package is to provide published module assignments between genes and transcriptional modules, as well as tools to analyse and visualize the modules.

### See Also

tmodHGtest, tmodUtest

## cell_signatures *Cell type signatures*

### Description

Cell type signatures

### Format

An object of class tmodGS

### Details

* CellMarker: Zhang X, Lan Y, Xu J, Quan F, Zhao E, Deng C, Luo T, Xu L, Liao G, Yan M, Ping Y. CellMarker: a manually curated resource of cell markers in human and mouse. Nucleic acids research. 2019 Jan 8;47(D1):D721-8.

* CIBERSORT: Newman AM, Liu CL, Green MR, Gentles AJ, Feng W, Xu Y, Hoang CD, Diehn M, Alizadeh AA. Robust enumeration of cell subsets from tissue expression profiles. Nature methods. 2015 May;12(5):453-7.

* PanglaoDB: Franzén O, Gan LM, Björkegren JL. PanglaoDB: a web server for exploration of mouse and human single-cell RNA sequencing data. Database. 2019 Jan 1;2019.

### Source

CIBERSORT, CellMarkers, PanglaoDB

## Examples

```
## to use cell signatures, type
data(cell_signatures)
data(vaccination)
gl <- vaccination$GeneName[ order(vaccination$qval.F.D1) ]
tmodCERNOtest(gl, mset=cell_signatures)
```

---

`check_tmod_gs` *Check an object of class tmodGS*

---

## Description

Check an object of class tmodGS

## Usage

```
check_tmod_gs(object)
```

## Arguments

| | |
|---|---|
| `object` | an object of class tmodGS |

---

`EgambiaResults` *Gene expression in TB patients and Healthy controls*

---

## Description

Gene expression in TB patients and Healthy controls

## Details

This data set has been constructed from the gene expression data set accessible in the Gene Expression Omnibus (GEO) at the accession number GSE28623. Ten healthy donors (NID, non-infected donors) and 10 tubercolosis patients (TB) have been randomly selected from the full data set, and top 25 genes with the highest IQR have been selected for further analysis. Genes without an Entrez gene (EG) identifier have likewise been omitted.

The Egambia object is a data frame. The first three columns are the gene symbol, gene name and Entrez gene (EG) identifier. The remaining columns correspond to the gene expression data.

## Examples

```
## Not run:
# The data set has been generated as follows:
# get the data set from GEO
library(GEOquery)
gambia <- getGEO("GSE28623")[[1]]

# Convert to limma and normalize
library(limma)
e <- new("EListRaw", list(E= exprs(gambia), genes=fData(gambia), targets= pData(gambia)))
e.bg <- backgroundCorrect(e, method= "normexp")
en <- normalizeBetweenArrays(e.bg, method= "q")
en <- avereps(en, ID=en$genes$NAME)
en <- en[ en$genes$CONTROL_TYPE == "FALSE", ]
en$targets$group <- factor(gsub(" whole blood RNA *", "", en$targets$description))

# Fill in Entrez Gene IDs
library(org.Hs.eg.db)
en$genes$EG <- ""
sel <- en$genes$REFSEQ %in% ls(org.Hs.egREFSEQ2EG)
en$genes$EG[sel] <- mget(as.character(en$genes$REFSEQ[sel]), org.Hs.egREFSEQ2EG)

# Filter by IQR and missing EG's
iqrs <- apply(en$E, 1, IQR)
en2 <- en[ iqrs > quantile(iqrs, 0.75) & en$genes$EG != "", ]

# Select 10 random samples from NID and TB groups
en2 <- en2[ , c(sample(which(en2$targets$group == "NID"), 10),
                sample(which(en2$targets$group == "TB"), 10)) ]
colnames(en2$E) <- en2$targets$group
Egambia <- cbind(en2$genes[ , c("GENE_SYMBOL", "GENE_NAME", "EG") ], en2$E)

## End(Not run)
```

---

`eigengene` *Calculate the eigengene of a module from a data set*

---

## Description

Calculate the eigengene of a module from a data set

## Usage

```
eigengene(x, g, mset = NULL, k = 1)
```

## Arguments

| | |
|---|---|
| x | data; genes in rows, samples in columns |
| g | genes – a vector gene IDs corresponding to annotation in modules |

| | |
|---|---|
| mset | – a module set; eigengenes will be calculated for each module in the set |
| k | which component defines the eigengene (default: 1) |

## Details

The eigengene of a module is here defined as the first principal component of a PCA on the gene expression of all genes from a module.

## Value

A numeric matrix with rows corresponding to modules. If there was not a sufficient number of genes in a module corresponding to the data set, the row will contain only NA's.

## Examples

```
data(Egambia)
data(tmod)
x <- Egambia[ , -c(1:3) ]
ifns <- tmod[ grep("[Ii]nterferon", tmod$gs$Title) ]
eigv <- eigengene(x, Egambia$GENE_SYMBOL, ifns)
plot(eigv["LI.M127", ], eigv["DC.M1.2",])

# which interferon modules are correlated
cor(eigv)
```

---

# evidencePlot *Create an evidence plot for a module*

---

## Description

Create an evidence plot for a module

## Usage

```
evidencePlot(
  l,
  m,
  mset = "all",
  rug = TRUE,
  roc = TRUE,
  filter = FALSE,
  unique = TRUE,
  add = FALSE,
  col = "black",
  col.rug = "#eeeeee",
  gene.labels = NULL,
  gene.colors = NULL,
  gene.lines = 1,
```

```
  gl.cex = 1,
  style = "roc",
  lwd = 1,
  lty = 1,
  rug.size = 0.2,
  legend = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| l | sorted list of HGNC gene identifiers |
| m | character vector of modules for which the plot should be created |
| mset | Which module set to use (see tmodUtest for details) |
| rug | if TRUE, draw a rug-plot beneath the ROC curve |
| roc | if TRUE, draw a ROC curve above the rug-plot |
| filter | if TRUE, genes not defined in the module set will be removed |
| unique | if TRUE, duplicates will be removed |
| add | if TRUE, the plot will be added to the existing plot |
| col | a character vector color to be used |
| col.rug | a character value specifying the color of the rug |
| gene.labels | if TRUE, gene names are shown; alternatively, a named character vector with gene labels to be shown, or NULL (default) for no labels (option evaluated only if rug is plotted) |
| gene.colors | NULL (default) or a character vectors indicating the color for each gene. Either a named vector or a vector with the same order of genes as ‘l‘. |
| gene.lines | a number or a vector of numbers; line width for marking the genes on the rug (default=1). If the vector is named, the names should be gene ids. |
| gl.cex | Text cex (magnification) for gene labels |
| style | "roc" for receiver-operator characteristic curve (default), and "gsea" for GSEA-style (Kaplan-Meier like plot) |
| lwd | line width (see par()) |
| lty | line type (see par()) |
| rug.size | fraction of the plot that should show the rug. If rug.size is 0, rug is not drawn. If rug.size is 1, ROC curve is not drawn. |
| legend | position of the legend. If NULL, no legend will be drawn |
| ... | Further parameters passed to the plotting function |

## Details

This function creates an evidence plot for a module, based on an ordered list of genes. By default, the plot shows the receiving operator characteristic (ROC) curve and a rug below, which indicates the distribution of the module genes in the sorted list.

Several styles of the evidence plot are possible: * roc (default): a receiver-operator characteristic like curve; the area under the curve corresponds to the effect size (AUC) * roc_absolute: same as above, but the values are not scaled by the total number of genes in a module * gsea * enrichment: the curve shows relative enrichment at the given position

## See Also

[tmod-package()], [hgEnrichmentPlot()]

## Examples

```
# artificially enriched list of genes
set.seed(123)
data(tmod)
bg <- sample(tmod$gv)
fg <- getGenes("LI.M127", as.list=TRUE)[[1]]
fg <- sample(c(fg, bg[1:1000]))
l <- unique(c(fg, bg))
evidencePlot(l, "LI.M127")
evidencePlot(l, filter=TRUE, "LI.M127")
```

---

# filterGS *Filter by genes belonging to a gene set from a data frame*

---

## Description

Filter a data frame or vector by genes belonging to a gene set

## Usage

```
filterGS(genes, gs, mset = "all")

showModule(x, genes, gs, mset = "all", extra = NULL)
```

## Arguments

| | |
|---|---|
| genes | a character vector with gene IDs |
| gs | a character vector corresponding to the IDs of the gene sets to be shown |
| mset | Module set to use; see "tmodUtest" for details |
| x | a data frame or a vector |
| extra | no longer used. |

## Details

filterGS filters a vector of gene IDs based on whether the IDs belong to a given set of gene sets, returning a logical vector.

The showModule function is deprecated and will be removed in future.

## Value

filterGS returns a logical vector of length equal to genes, with TRUE indicating that the given gene is a member of the gene sets in ‘gs‘.

## Examples

```
data(Egambia)
## LI.M127 - type I interferon response
sel <- filterGS("LI.M127", Egambia$GENE_SYMBOL)
head(Egambia[sel, ])
```

---

# getGenes *Get genes belonging to a gene set*

---

## Description

Get genes belonging to a gene set

## Usage

```
getGenes(gs = NULL, genes = NULL, fg = NULL, mset = "all", as.list = FALSE)
```

## Arguments

| | |
|---|---|
| gs | gene set IDs; if NULL, returns all genes from all gene sets |
| genes | character vector with gene IDs. If not NULL, only genes from this parameter will be considered. |
| fg | genes which are in the foreground set |
| mset | gene set to use (default: all tmod gene sets) |
| as.list | should a list of genes rather than a data frame be returned |

## Details

Create a data frame mapping each module to a comma separated list of genes. If genelist is provided, then only genes in that list will be shown. An optional column, "fg" informs which genes are in the "foreground" data set.

## Value

data frame containing module to gene mapping, or a list (if as.list == TRUE

## getModuleMembers *Return the contents of a gene set*

### Description

Return the contents of a gene set

### Usage

```
getModuleMembers(x, mset = "all")
```

### Arguments

| | |
|---|---|
| x | a character vector of gene set names |
| mset | optional, a gene set collection |

### Details

This function returns the selected gene sets from a collection.

### Value

A list of gene sets

### Examples

```
# show the interferon related modules
getModuleMembers(c("LI.M127", "LI.M158.0", "LI.M158.0"))
getModuleMembers("LI.M127")
```

## ggEvidencePlot *Create an evidence plot for a module (ggplot2 version)*

### Description

Create an evidence plot for a module (ggplot2 version)

### Usage

```
ggEvidencePlot(
  l,
  m,
  mset = NULL,
  filter = FALSE,
  unique = TRUE,
  gene.labels = NULL,
  gene.colors = NULL
)
```

### Arguments

| | |
|---|---|
| l | sorted list of HGNC gene identifiers |
| m | character vector of modules for which the plot should be created |
| mset | Which module set to use (see tmodUtest for details) |
| filter | if TRUE, genes not defined in the module set will be removed |
| unique | if TRUE, duplicates will be removed |
| gene.labels | if TRUE, gene names are shown; alternatively, a named character vector with gene labels to be shown, or NULL (default) for no labels (option evaluated only if rug is plotted) |
| gene.colors | NULL (default) or a character vectors indicating the color for each gene. Either a named vector or a vector with the same order of genes as 'l'. |

---

## ggPanelplot *Create a tmod panel plot using ggplot*

---

### Description

Create a tmod panel plot using ggplot

### Usage

```
ggPanelplot(
  res,
  sgenes = NULL,
  auc_thr = 0.5,
  q_thr = 0.05,
  filter_row_q = 0.01,
  filter_row_auc = 0.65,
  q_cutoff = 1e-12,
  cluster = TRUE,
  id_order = NULL,
  effect_size = "auto",
  colors = c("red", "grey", "blue"),
  label_angle = 45,
  add_ids = TRUE,
  mset = NULL
)
```

### Arguments

| | |
|---|---|
| res | a list with tmod results (each element of the list is a data frame returned by a tmod test function) |
| sgenes | a list with summaries of significantly DE genes by gene set. Each a element of the list is a matrix returned by tmodDecideTests. If NULL, the bars on the plot will be monochromatic. |

| | |
|---|---|
| auc_thr | gene sets enrichments with AUC (or other effect size) below ‘auc_thr‘ will not be shown |
| q_thr | gene sets enrichments with q (adjusted P value) above ‘q_thr‘ will not be shown |
| filter_row_q | Gene sets will only be shown if at least in one contrast q is smaller than ‘filter_row_q‘ |
| filter_row_auc | Gene sets will only be shown if at least in one contrast AUC (or other effect size if specified) is larger than ‘filter_row_auc‘ |
| q_cutoff | Any q value below ‘q_cutoff‘ will be considered equal to ‘q_cutoff‘ |
| cluster | whether to cluster the IDs by similarity |
| id_order | character vector specifying the order of IDs to be shown. This needs not to contain all IDs shown, but whatever IDs are in this vector, they will be shown on top in the given order. |
| effect_size | name of the column which contains the effect sizes; by default, the name of this column is taken from the "effect_size" attribute of the first result table. |
| colors | character vector with at least 1 (when sgenes is NULL) or 3 (when sgenes is not NULL) elements |
| label_angle | The angle at which column labels are shown |
| add_ids | add IDs of gene sets to their titles on the plot |
| mset | an object of the type ’tmodGS’. If the option ‘cluster‘ is TRUE, the mset object is used to cluster the gene sets. By default, the built-in transcription modules are used. See details. |

## Details

Panel plot is a kind of a heatmap. This is the most compact way of representing the results of several gene set enrichment analyses. Each row of a panel plot shows the result for one gene set, and each column shows corresponds to one analysis. For example, if one tests gene set enrichment for a number of different contrasts, then each contrast will be represented in a separate column.

Each cell of a panel plot shows both the effect size and the p-value. The p-value is encoded as transparency: the enrichments with a lower p-value have stronger colors. The size of the bar corresponds to the effect size, however it is defined. For example, in case of the tmodCERNOtest, tmodZtest or tmodUtest it is the area under curve, AUC.

In addition, the bars may also encode information about the number of up- or down-regulated genes. For this, an object must be created using the function tmodDecideTests. This object provides information about which genes in a particular gene set are regulated in which direction.

The order of the gene sets displayed is, by default, determined by clustering the gene sets based on their overlaps. For this to work, ggPanelplot must know about what genes are contained in which gene sets. This is provided by the parameter ‘mset‘. By default (when mset is NULL) this is the built-in list of gene sets. If, however, the results of gene set enrichment come from a different set of gene sets, you need to specify it with the mset parameter. See Examples.

## Value

The object returned is a ggplot2 object which can be further modified the usual way.

## See Also

[tmodDecideTests()], [tmodPanelPlot()]

## Examples

```
## prepare a set of results
data(Egambia)
genes <- Egambia$GENE_SYMBOL
exprs <- Egambia[ , -1:-4 ]
group <- gsub("\\..*", "", colnames(exprs))
## test differential expression using limma
design <- cbind(Intercept=rep(1, 30), TB=rep(c(0,1), each= 15))
## Not run:
library(limma)
fit <- eBayes( lmFit(Egambia[,-c(1:3)], design))
tt <- topTable(fit, coef=2, number=Inf, genelist=Egambia[,1:3] )
res <- tmodCERNOtest(tt$GENE_SYMBOL)
## show the results using a panel plot
ggPanelplot(list(limma=res))
## add information about the significant genes
sgenes <- tmodDecideTests(tt$GENE_SYMBOL, lfc=tt$logFC, pval=tt$adj.P.Val)
names(sgenes) <- "limma"
ggPanelplot(list(limma=res), sgenes=sgenes)
## we will now compare the results of enrichments for different types of
## differential expression tests on the data
res_utest <- apply(exprs, 1, function(x) wilcox.test(x ~ group)$p.value)
res_ttest <- apply(exprs, 1, function(x) t.test(x ~ group)$p.value)
## Calculate the gene set enrichment analysis results for each of the
## different types of tests
res_tmod <- list()
res_tmod$limma <- res
res_tmod$utest <- tmodCERNOtest(genes[order(res_utest)])
res_tmod$ttest <- tmodCERNOtest(genes[order(res_ttest)])
ggPanelplot(res_tmod)
## Using the `mset` parameter
## First, we generate results using a different set of gene sets
data(cell_signatures)
res_cs <- tmodCERNOtest(tt$GENE_SYMBOL, mset=cell_signatures)
## the following will triger a warning that no clustering is possible
## because ggPanelplot doesn't have the information about the gene set
## contents
ggPanelplot(list(res=res_cs))
## if we use the mset parameter, clustering is available
ggPanelplot(list(res=res_cs), mset=cell_signatures)

## End(Not run)
```

---

`hgEnrichmentPlot` *Create a visualisation of enrichment*

---

## Description

Create a visualisation of enrichment

## Usage

```
hgEnrichmentPlot(fg, bg, m, mset = "all", ...)
```

## Arguments

| | |
|---|---|
| fg | the foreground set of genes |
| bg | the background set of genes (gene universe) |
| m | gene set for which the plot should be created |
| mset | Which module set to use (see tmodUtest for details) |
| ... | additional parameters to be passed to the plotting function |

## Details

This functions creates a barplot visualizing the enrichment of a module in the foreground (fg) set as compared to the background (bg) set. It is the counterpart

## See Also

tmod-package, [evidencePlot()]

## Examples

```
set.seed(123)
data(tmod)
bg <- tmod$gv
fg <- getGenes("LI.M127", as.list=TRUE)[[1]]
fg <- sample(c(fg, bg[1:100]))
hgEnrichmentPlot(fg, bg, "LI.M127")
```

---

# makeTmodFromDataFrame *Convert a data frame to a tmod object*

---

## Description

Convert a data frame to a tmod object

## Usage

```
makeTmodFromDataFrame(
  df,
  feature_col = 1,
  module_col = 2,
  title_col = NULL,
  extra_module_cols = NULL,
  extra_gene_cols = NULL
)
```

## Arguments

| | |
|---|---|
| df | A data frame |
| feature_col | Which column contains the feature (gene) IDs |
| module_col | Which column contains the module (gene set) IDs |
| title_col | Description of the modules (if NULL, the description will be taken from the module_col) |
| extra_module_cols | Additional columns to include in the module data frame |
| extra_gene_cols | Additional gene columns to include in the genes data frame |

## Details

'makeTmodFromFeatureDataFrame' converts mapping information from features (genes) to modules (gene sets). The data frame has a row for each feature-module pair.

'makeTmodFromModuleDataFrame' converts mapping information from features (genes) to modules (gene sets). The data frame has a row for each module, and all gene IDs corresponding to a module are stored as a comma separated string, e.g.

Vice versa, 'tmod2DataFrame' converts a tmod object to a data frame.

## Value

A tmod object

## See Also

makeTmodGS, makeTmodGS

## Examples

```
df <- data.frame(
gene_id=LETTERS[1:10],
geneset_id=rep(letters[1:2], each=5),
geneset_description=rep(paste0("Gene set ", letters[1:2]), each=5))
res <- makeTmodFromDataFrame(df,
  feature_col="gene_id",
  module_col="geneset_id",
  title_col="geneset_description")
```

makeTmodGS *S3 class for tmod gene set collections*

## Description

S3 class for tmod gene set collections

## Usage

```
makeTmodGS(gs2gene, gs = NULL, weights = NULL, info = NULL)

makeTmod(modules, modules2genes, genes2modules = NULL, genes = NULL)

as_tmodGS(x, check_sanity = FALSE)

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

## S3 method for class 'tmodGS'
length(x)

## S3 method for class 'tmodGS'
x[i]

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

## Arguments

| | |
|---|---|
| gs2gene, modules2genes | A list with module IDs as names. Each member of the list is a character vector with IDs of genes contained in that module |
| gs, modules | [Optional] A data frame with at least columns ID and Title |
| weights | [Optional] a named numeric vector of weights for each gene set |
| info | [Optional] a list containing meta-information about the gene set collection |
| genes2modules, genes | Ignored |
| x | a tmodGS or tmod object |
| check_sanity | whether the tmodGS object should be tested for correctness |
| ... | further arguments passed to ‘print()‘ |
| i | indices specifying elements to extract or replace |

## Details

An object of class tmod contains the gene set annotations (‘tmod$gs‘), a character vector of gene identifiers (‘tmod$gv‘) and a mapping between gene sets and gene identifiers (‘tmod$gs2gv‘). Optionally, a vector of numeric weights of the same length as ‘gs2gv‘ may be provided (‘tmod$weights‘).

Furthermore, it may contain additional information about the gene set (‘tmod$info‘).

‘tmod$gs‘ is a data frame which must contain the column "ID". Additional optional columns ‘Title‘ and ‘Description‘ are recognized by some functions. Any further columns may contain additional information on the gene sets. The number of rows of that data frame is equal to the number of gene sets in a gene set collection.

Each element of the tmod$g2m list corresponds to the respective row of the ‘tmod$gs‘ data frame. Each element is an integer vector containing the positions of the gene identifiers in the ‘tmod$gv‘ character vector.

Objects of class tmodGS should be constructed by calling the function makeTmodGS(). This function check the validity and consistency of the provided objects.

The makeTmod function remains for compatibility with previous versions of the package. It produces the objects of the new class tmodGS, however.

See the package vignette for more on constructing custom module sets.

## See Also

tmod-data

## Examples

```
# A minimal example
gs <- data.frame(ID=letters[1:3], Title=LETTERS[1:3])
gs2gv <- list(a=c("g1", "g2"), b=c("g3", "g4"), c=c("g1", "g2", "g4"))
mymset <- makeTmodGS(gs2gene=gs2gv, gs=gs)
str(mymset)
```

---

# modCorPlot *Plot a correlation heatmap for modules*

---

## Description

Plot a correlation heatmap for modules

## Usage

```
modCorPlot(
  modules,
  mset = NULL,
  heatmap_func = pheatmap,
  labels = NULL,
  stat = "jaccard",
```

```
  upper.cutoff = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| modules | either a character vector with module IDs from mset, or a list which contains the module members |
| mset | Which module set to use. Either a character vector ("LI", "DC" or "all", default: all) or an object of class tmod (see "Custom module definitions" below) |
| heatmap_func | function drawing the heatmap |
| labels | Labels for the modules (if NULL, labels will be retrieved from ‘mset‘) |
| stat | Type of statistics to return. "jaccard": Jaccard index (default); "number": number of common genes; "soerensen": Soerensen-Dice coefficient; "overlap": Szymkiewicz-Simpson coefficient. |
| upper.cutoff | Specify upper cutoff for the color palette |
| ... | Any further parameters are passed to the heatmap function (by default, [pheatmap()]. |

### Value

Returns the return value of heatmap_func (by default, a pheatmap object).

---

## modcors *Module correlation*

---

### Description

Calculate the correlation between modules

### Usage

```
modcors(x, g, mset = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | a data set, with variables (e.g. genes) in rows and samples in columns |
| g | a vector of variable idenitifiers which correspond to the definition of modules |
| mset | a module set |
| ... | any further parameters will be passed to the cor() function |

### Details

The correlation between modules are defined as correlation coefficient between the modules eigengenes. These are based on a particular gene expression data set.

This function is a simple wrapper combining eigengene() and cor().

### Value

a matrix of module correlation coefficients

---

`modGroups` *Find group of modules*

---

### Description

Find group of modules based on shared genes

### Usage

```
modGroups(modules, mset = NULL, min.overlap = 2, stat = "number")
```

### Arguments

| | |
|---|---|
| `modules` | Either a list of modules or character vector. |
| `mset` | Which module set to use. Either a character vector ("LI", "DC" or "all", default: all) or an object of class tmod (see "Custom module definitions" below) |
| `min.overlap` | Minimum number of overlapping items if stat == number, minimum jaccard index if stat == jaccard etc. |
| `stat` | Type of statistics to return. "jaccard": Jaccard index (default); "number": number of common genes; "soerensen": Soerensen-Dice coefficient; "overlap": Szymkiewicz-Simpson coefficient. |

### Details

Split the modules into groups based on the overlapping items.

The first argument, modules, is either a character vector of module identifiers from ‘mset‘ (NULL mset indicates the default mset of tmod) or a list. If it is a list, then each element is assumed to be a character vector with module IDs.

### Examples

```
mymods <- list(A=c(1, 2, 3), B=c(2, 3, 4, 5), C=c(5, 6, 7))
modGroups(mymods)
```

## modjaccard *Jaccard index for modules*

### Description

Jaccard index for modules

### Usage

```
modjaccard(mset = NULL, g = NULL)
```

### Arguments

| | |
|---|---|
| mset | set of modules for which to calculate the Jaccard index. If NULL, the default tmod module set will be used. |
| g | a list of genes. If NULL, the list of genes from the mset will be used. |

### Details

For each pair of modules in mset, calculate the Jacard index between these modules.

### Value

matrix with Jaccard index for each pair of modules in mset

## modmetabo *Modules for metabolic profiling*

### Description

Feature and data sets for metabolic profiling

### Details

The module set "modmetabo" can be used with tmod to analyse metabolic profiling data. The clusters defined in this set are based on hierarchical clustering of metabolic compounds from human serum and have been published in a paper on metabolic profiling in tuberculosis by Weiner et al. (2012).

For an example analysis, "tbmprof" is a data set containing metabolic profiles of serum isolated from tuberculosis (TB) patients and healthy individuals. The tbmprof is a data frame containing observations in rows and metabolite id's (corresponding to the id's in the modmetabo object). See examples below.

## References

Weiner 3rd J, Parida SK, Maertzdorf J, Black GF, Repsilber D, Telaar A, Mohney RP, Arndt-Sullivan C, Ganoza CA, Fae KC, Walzl G. Biomarkers of inflammation, immunosuppression and stress are revealed by metabolomic profiling of tuberculosis patients. PloS one. 2012 Jul 23;7(7):e40221.

## See Also

tmod-data

## Examples

```
data(modmetabo)  # module definitions
data(tbmprof)    # example data set
ids <- rownames(tbmprof)
tb  <- factor(gsub("\\..*", "", ids))

## use Wilcoxon test to calculate significant differences
wcx <- apply(tbmprof, 2, function(x) wilcox.test(x ~ tb)$p.value)
wcx <- sort(wcx)
tmodCERNOtest(names(wcx), mset=modmetabo)
```

---

# modOverlaps *Calculate overlaps of the modules*

---

## Description

Calculate overlaps of the modules

## Usage

```
modOverlaps(modules = NULL, mset = NULL, stat = "jaccard")
```

## Arguments

| | |
|---|---|
| modules | either a character vector with module IDs from mset, or a list which contains the module members |
| mset | Which module set to use. Either a character vector ("LI", "DC" or "all", default: all) or an object of class tmod (see "Custom module definitions" below) |
| stat | Type of statistics to return. "jaccard": Jaccard index (default); "number": number of common genes; "soerensen": Soerensen-Dice coefficient; "overlap": Szymkiewicz-Simpson coefficient. |

### Details

For a set of modules (aka gene sets) determine the similarity between these. You can run modOverlaps either on a character vector of module IDs or on a list. In the first case, the module elements are taken from 'mset', and if that is NULL, from the default tmod module set.

Alternatively, you can provide a list in which each element is a character vector. In this case, the names of the list are the module IDs, and the character vectors contain the associated elements.

The different statistics available are: * "number": total number of common genes (size of the overlap) * "jaccard": Jaccard index, i.e. $\frac{|A \cap B|}{|A \cup B|}$ (number of common elements divided by the total number of unique elements); * "soerensen": Soerensen-Dice coefficient, defined as $\frac{2 \cdot |A \cap B|}{|A|+|B|}$ – number of common genes in relation to the total number of elements (divided by two, such that the maximum is 1) (number of common elements divided by the average size of both gene sets) * "overlap": Szymkiewicz-Simpson coefficient, defined as $\frac{|A \cap B|}{\min(|A|,|B|)}$ – this is the number of common genes scaled by the size of the smaller of the two gene sets (number of common elements divided by the size of the smaller gene set)

---

## pcaplot *Plot a PCA object returned by prcomp*

---

### Description

Plot a PCA object returned by prcomp

### Usage

```
pcaplot(
  pca,
  components = 1:2,
  group = NULL,
  col = NULL,
  pch = 19,
  cex = 2,
  legend = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| pca | PCA object returned by prcomp |
| components | a vector of length two indicating the components to plot |
| group | a factor determining shapes of the points to show (unless overriden by pch=...) |
| col | Color for plotting (default: internal palette) |
| pch | Type of character to plot (default: 19) |
| cex | size of the symbols used for plotting |
| legend | draw a legend? If legend is a position (eg. "topright"), then a legend is drawn. If NULL or if the group parameter is NULL, then not. |
| ... | any further parameters will be passed to the plot() function (e.g. col, cex, ...) |

### Details

This is a simplistic function. A much better way is to use the pca2d function from the pca3d package.

### Value

If group is NULL, then NULL; else a data frame containing colors and shapes matching each group

---

`pvalEffectPlot` *Create an effect size / p-value plot*

---

### Description

Create a heatmap-like plot showing information about both effect size and p-values.

### Usage

```
pvalEffectPlot(
  e,
  p,
  pval.thr = 0.01,
  pval.cutoff = 1e-06,
  row.labels = NULL,
  col.labels = NULL,
  plot.func = NULL,
  grid = "at",
  grid.color = "#33333333",
  plot.cex = 1,
  text.cex = 1,
  col.labels.style = "top",
  symmetrical = FALSE,
  legend.style = "auto",
  min.e = NULL,
  max.e = NULL
)
```

### Arguments

| | |
|---|---|
| `e` | matrix with effect sizes |
| `p` | matrix with probabilities |
| `pval.thr` | The p-value must be this or lower in order for a test result to be visualized |
| `pval.cutoff` | On visual scale, all p-values below pval.cutoff will be replaced by pval.cutoff |
| `row.labels` | Labels for the modules. This must be a named vector, with module IDs as vector names. If NULL, module titles from the analyses results will be used. |
| `col.labels` | Labels for the columns. If NULL, names of the elements of the list x will be used. |

| | |
|---|---|
| plot.func | Optionally, a function to be used to draw the dots. See "Details" |
| grid | Style of a light-grey grid to be plotted; can be "none", "at" and "between" |
| grid.color | Color of the grid to be plotted (default: light grey) |
| plot.cex | a numerical value giving the amount by which the plot symbols will be magnified |
| text.cex | a numerical value giving the amount by which the plot text will be magnified, or a vector containing three cex values for row labels, column labels and legend, respectively |
| col.labels.style | Style of column names: "top" (default), "bottom", "both", "none" |
| symmetrical | effect sizes are distributed symmetrically around 0 (default: FALSE) |
| legend.style | Style of the legend: "auto" – automatic; "broad": pval legend side by side with effect size legend; "tall": effect size legend above pval legend; "none" – no legend. |
| min.e, max.e | scale limits for the effect size |

## Details

pvalEffectPlot shows a heatmap-like plot. Each row corresponds to one series of tests (e.g. one module), and each column corresponds to the time points or conditions for which a given analysis was run. Each significant result is shown as a red dot. Size of the dot corresponds to the effect size (or any arbitrary value), and intensity of the color corresponds to the log10 of p-value.

Just like a heatmap corresponds to a single numeric matrix, the pvalue / effect plot corresponds to two matrices: one with the effect size, and another one with the p-values. Each cell in the matrix corresponds to the results of a single statistical test.

For example, a number of genes or transcriptional modules might be tested for differential expression or enrichment, respectively, in several conditions.

By default, each test outcome is represented by a dot of varying size and color. Alternatively, a function may be specified with the parameter 'plot.func'. It will be called for each test result to be drawn. The plot.func function must take the following arguments:

- row, coleither row / column number or the id of the row / column to plot; NULL if drawing legend
- x, yuser coordinates of the result to visualize
- w, hwidth and height of the item to plot
- eEnrichment – a relative value between 0 and 1, where 0 is the minimum and 1 is the maximum enrichment found
- pP-value – an absolute value between 0 and 1

For the purposes of drawing a legend, the function must accept NULL p-value or a NULL enrichment parameter.

## Value

Invisibly returns a NULL value.

showGene *A combined beeswarm / boxplot*

## Description

A combined beeswarm / boxplot

## Usage

```
showGene(
  data,
  group,
  main = "",
  pch = 19,
  xlab = "",
  ylab = "log2 expression",
  las = 2,
  pwcol = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| data | a vector of numeric values to be plotted |
| group | factor describing the groups |
| main | title of the plot |
| pch | character to plot the points |
| xlab, ylab | x and y axis labels |
| las | see par() |
| pwcol | colors of the points (see beeswarm) |
| ... | any additional parameters to be passed to the beeswarm command |

## Details

This is just a simple wrapper around the beeswarm() and boxplot() commands.

## Examples

```
data(Egambia)
E <- as.matrix(Egambia[,-c(1:3)])
showGene(E["20799",], rep(c("CTRL", "TB"), each=15))
```

simplePie *Simple Pie Chart*

## Description

The simplePie function draws a simple pie chart at specified coordinates with specified width, height and color. The simpleRug function draws a corresponding rug plot, while simpleBoxpie creates a "rectangular pie chart" that is considered to be better legible than the regular pie.

## Usage

```
simplePie(x, y, w, h, v, col, res = 100, border = NA)

simpleRug(x, y, w, h, v, col, border = NULL)

simpleBoxpie(x, y, w, h, v, col, border = NA, grid = 3)
```

## Arguments

| | |
|---|---|
| x, y | coordinates at which to draw the plot |
| w, h | width and height of the plot |
| v | sizes of the slices |
| col | colors of the slices |
| res | resolution (number of polygon edges in a full circle) |
| border | color of the border. Use NA (default) or NULL for no border |
| grid | boxpie only: the grid over which the areas are distributed. Should be roughly equal to the number of areas shown. |

## Details

simplePie() draws a pie chart with width w and height h at coordinates (x,y). The size of the slices is taken from the numeric vector v, and their color from the character vector col.

## Examples

```
# demonstration of the three widgets
plot.new()
par(usr=c(0,3,0,3))
x <- c(7, 5, 11)
col <- tmodPal()
b <- "black"
simpleRug(0.5, 1.5, 0.8, 0.8, v=x, col=col, border=b)
simplePie(1.5, 1.5, 0.8, 0.8, v=x, col=col, border=b)
simpleBoxpie(2.5, 1.5, 0.8, 0.8, v=x, col=col, border=b)

# using pie as plotting symbol
```

```
plot(NULL, xlim=1:2, ylim=1:2, xlab="", ylab="")
col <- c("#cc000099", "#0000cc99")
for(i in 1:125) {
  x <- runif(1) + 1
  y <- runif(1) + 1
  simplePie( x, y, 0.05, 0.05, c(x,y), col)
}

# square filled with box pies
n <- 10
w <- h <- 1/(n+1)
plot.new()
for(i in 1:n) for(j in 1:n)
 simpleBoxpie(1/n*(i-1/2), 1/n*(j-1/2), w, h,
 v=runif(3), col=tmodPal())
```

---

tmod-data *Default gene expression module data*

---

## Description

Gene expression module data from Chaussabel et al. (2008) and Li et al. (2014)

## Details

The tmod package includes one data set of class tmod which can be loaded with data(tmod). This data set is derived from two studies (see package vignette for details). By default, enrichment analysis with tmod uses this data set; however, it is not loaded into user workspace by default.

## References

Chaussabel, Damien, Charles Quinn, Jing Shen, Pinakeen Patel, Casey Glaser, Nicole Baldwin, Dorothee Stichweh, et al. 2008. "A Modular Analysis Framework for Blood Genomics Studies: Application to Systemic Lupus Erythematosus." Immunity 29(1):150-64.

Li, Shuzhao, Nadine Rouphael, Sai Duraisingham, Sandra Romero-Steiner, Scott Presnell, Carl Davis, Daniel S Schmidt, et al. 2014. "Molecular Signatures of Antibody Responses Derived from a Systems Biology Study of Five Human Vaccines." Nature Immunology 15(2):195-204.

## See Also

tmod-class, modmetabo

## Examples

```
# list of first 10 modules
data(tmod)
tmod
tmod$MODULES[1:10, ]
tmod[1:10]
```

## tmod2DataFrame *Convert a tmod module set into a data frame*

### Description

Convert a tmod module set into a data frame

### Usage

```
tmod2DataFrame(
  mset,
  rows = "modules",
  module_col = "module_id",
  feature_col = "feature_id",
  sep = ","
)
```

### Arguments

| | |
|---|---|
| mset | a tmod object (e.g. generated by makeTmod) |
| rows | if "modules", then there will be a row corresponding to each module (gene set); if "features", then there will be a row corresponding to each gene. |
| module_col | Name of the column with module (gene set) IDs |
| feature_col | Name of the column with feature (gene) IDs |
| sep | separator used to collate module IDs (if rows=="features") or feature IDs (if rows=="modules") |

### See Also

makeTmodGS, makeTmod

## tmod2tmodGS *Convert the old tmod objects to the tmodGS objects*

### Description

Convert the old tmod objects to the tmodGS objects

### Usage

```
tmod2tmodGS(x)
```

### Arguments

| | |
|---|---|
| x | an object of class tmod |

### Details

The old tmod representation was very inefficient. This function converts the objects to a new representation which is both allowing faster computations and more memory efficient.

### Value

Returns an object of class tmodGS.

---

`tmodAUC` *Calculate AUC*

---

### Description

Calculate AUC

### Usage

```
tmodAUC(
  l,
  ranks,
  modules = NULL,
  stat = "AUC",
  recalculate.ranks = TRUE,
  filter = FALSE,
  mset = "all"
)
```

### Arguments

| | |
|---|---|
| `l` | List of gene names corresponding to rows from the ranks matrix |
| `ranks` | a matrix with ranks, where columns correspond to samples and rows to genes from the l list |
| `modules` | optional list of modules for which to make the test |
| `stat` | Which statistics to generate. Default: AUC |
| `recalculate.ranks` | Filtering and removing duplicates will also remove ranks, so that they should be recalculated. Use FALSE if you don't want this behavior. If unsure, stay with TRUE |
| `filter` | Remove gene names which have no module assignments |
| `mset` | Which module set to use. "LI", "DC" or "all" (default: "all") |

## Details

tmodAUC calculates the AUC and U statistics. The main purpose of this function is the use in randomization tests. While tmodCERNOtest and tmodUtest both calculate, for each module, the enrichment in a single sorted list of genes, tmodAUC takes any number of such sorted lists. Or, actually, sortings – vectors with ranks of the genes in each replicate.

Note that the input for this function is different from tmodUtest and related functions: the ordering of l and the matrix ranks does not matter, as long as the matrix ranks contains the actual rankings. Each column in the ranks matrix is treated as a separate sample.

Also, the ‘nodups‘ parameter which is available (and TRUE by default) for other tests cannot be used here. This means that the AUCs calculated here might be slightly different from the AUCs calculated with default parameters in tests such as the [tmodCERNOtest()]. Use ‘nodups=FALSE‘ with [tmodCERNOtest()] to obtain identical results as with ‘tmodAUC‘.

## Value

A matrix with the same number of columns as "ranks" and as many rows as there were modules to be tested.

## See Also

tmod-package

## Examples

```
data(tmod)
l <- tmod_ids(tmod)
ranks <- 1:length(l)
res <- tmodAUC(l, ranks)
head(res)
```

---

# tmodDecideTests *Count the Up- or Down-regulated genes per module*

---

## Description

For each module in a set, calculate the number of genes which are in that module and which are significantly up- or down-regulated.

## Usage

```
tmodDecideTests(
  g,
  lfc = NULL,
  pval = NULL,
  lfc.thr = 0.5,
  pval.thr = 0.05,
  labels = NULL,
```

```
  filter.unknown = FALSE,
  mset = "all"
)
```

## Arguments

| | |
|---|---|
| g | a character vector with gene symbols |
| lfc | a numeric vector or a matrix with log fold changes |
| pval | a numeric vector or a matrix with p-values. Must have the same dimensions as lfc |
| lfc.thr | log fold change threshold |
| pval.thr | p-value threshold |
| labels | Names of the comparisons. Either NULL or a character vector of length equal to the number of columns in lfc and pval. |
| filter.unknown | If TRUE, modules with no annotation will be omitted |
| mset | Which module set to use. Either a character vector ("LI", "DC" or "all", default: LI) or a list (see "Custom module definitions" below) |

## Details

This function can be used to decide whether a module, as a whole, is up- or down regulated. For each module, it calculates the number of genes which are up-, down- or not regulated. A gene is considered to be up- regulated if the associated p-value is smaller than pval.thr and the associated log fold change is greater than lfc.thr. A gene is considered to be down- regulated if the associated p-value is smaller than pval.thr and the associated log fold change is smaller than lfc.thr.

Note that unlike decideTests from limma, tmodDecideTests does not correct the p-values for multiple testing – therefore, the p-values should already be corrected.

## Value

A list with as many elements as there were comparisons (columns in lfc and pval). Each element of the list is a data frame with the columns "Down", "Zero" and "Up" giving the number of the down-, not- and up-regulated genes respectively. Rows of the data frame correspond to module IDs.

## See Also

tmodSummary, tmodPanelPlot, tmodDecideTestsLimma

`tmodImportMSigDB` *Import data from MSigDB*

## Description

Import data from an MSigDB file in either XML or GMT format

## Usage

```
tmodImportMSigDB(
  file = NULL,
  format = "xml",
  organism = "Homo sapiens",
 fields = c("STANDARD_NAME", "CATEGORY_CODE", "SUB_CATEGORY_CODE", "EXACT_SOURCE",
    "EXTERNAL_DETAILS_URL")
)
```

## Arguments

| | |
|---|---|
| `file` | The name of the file to parse |
| `format` | Format (either "xml" or "gmt") |
| `organism` | Select the organism to use. Use "all" for all organisms in the file (only for "xml" format; default: "Homo sapiens") |
| `fields` | Which fields to import to the MODULES data frame (only for "xml" format) |

## Details

This command parses a file from MSigDB. Both XML and the MSigDB-specific "GMT" format are supported (however, the latter is discouraged, as it contains less information).

## Value

A tmod object

## Examples

```
## Not run:
## First, download the file "msigdb_v7.5.1.xml"
## from http://www.broadinstitute.org/gsea/downloads.jsp
msig <- tmodImportMSigDB("msigdb_v7.5.1.xml")

## End(Not run)
```

## tmodLEA *Leading Edge Analysis*

### Description

For each module, return a list of genes on the leading edge

### Usage

```
tmodLEA(l, modules, mset = "all", nodups = TRUE, filter = FALSE)
```

### Arguments

| | |
|---|---|
| l | list of genes |
| modules | character vector of module IDs for which to run the LEA |
| mset | Which module set to use. Either a character vector ("LI", "DC" or "all", default: LI) or an object of class tmod |
| nodups | Remove duplicate gene names in l and corresponding rows from ranks |
| filter | Remove gene names which have no module assignments |

### Details

Given a vector of ordered gene identifiers and a vector of module IDs, for each module, return the genes which are on the up-slope of the GSEA-style evidence plot. That is, return the genes that are driving the enrichment.

## tmodLEASummary *Summary stats of a leading edge analysis*

### Description

Summary stats of a leading edge analysis

### Usage

```
tmodLEASummary(lea, genes = FALSE, labels = NULL, mset = NULL)
```

### Arguments

| | |
|---|---|
| lea | result of ‘tmodLEA‘ |
| genes | if TRUE, the gene identifiers of the leading edge (joined by commas) will be appended. |
| labels | labels to add to the result; if NULL, the labels will be taken from ‘mset‘ |
| mset | Which module set to use. Either a character vector ("LI", "DC" or "all", default: all) or an object of class tmod (see "Custom module definitions" below) |

## Value

data frame with summary stats

---

`tmodLimmaDecideTests` *Up- and down-regulated genes in modules based on limma object*

---

## Description

For each module in mset and for each coefficient in f$coefficients, this function calculates the numbers of significantly up- and down-regulated genes.

## Usage

```
tmodLimmaDecideTests(
  f,
  genes,
  lfc.thr = 0.5,
  pval.thr = 0.05,
  filter.unknown = FALSE,
  adjust.method = "BH",
  mset = "all"
)
```

## Arguments

| | |
|---|---|
| `f` | result of linear model fit produced by limma functions lmFit and eBayes |
| `genes` | Either the name of the column in f$genes which contains the gene symbols corresponding to the gene set collection used, or a character vector with gene symbols |
| `lfc.thr` | log fold change threshold |
| `pval.thr` | p-value threshold |
| `filter.unknown` | If TRUE, modules with no annotation will be omitted |
| `adjust.method` | method used to adjust the p-values for multiple testing. See p.adjust(). Default:BH. |
| `mset` | Which module set to use (see tmodUtest for details) |

## Details

For an f object returned by eBayes(), tmodLimmaDecideTests considers every coefficient in this model (every column of f$coefficients). For each such coefficient, tmodLimmaDecideTests calculates, for each module, the number of genes which are up- or down-regulated.

In short, tmodLimmaDecideTests is the equivalent of tmodDecideTests, but for limma objects returned by eBayes().

## Value

A list with as many elements as there were coefficients in f. Each element of the list is a data frame with the columns "Down", "Zero" and "Up" giving the number of the down-, not- and up-regulated genes respectively. Rows of the data frame correspond to module IDs. The object can directly be used in tmodPanelPlot as the pie parameter.

## See Also

tmodDecideTests, tmodLimmaTest, tmodPanelPlot

## Examples

```
## Not run:
data(Egambia)
design <- cbind(Intercept=rep(1, 30), TB=rep(c(0,1), each= 15))
if(require(limma)) {
  fit <- eBayes( lmFit(Egambia[,-c(1:3)], design))
  ret <- tmodLimmaTest(fit, Egambia$GENE_SYMBOL)
  pie <- tmodLimmaDecideTests(fit, Egambia$GENE_SYMBOL)
  tmodPanelPlot(ret, pie=pie)
}

## End(Not run)
```

---

# tmodLimmaTest *Run tmod enrichment tests directly on a limma object*

---

## Description

Order the genes according to each of the coefficient found in a limma object and run an enrichment test on the ordered list.

## Usage

```
tmodLimmaTest(
  f,
  genes,
  sort.by = "msd",
  tmodFunc = tmodCERNOtest,
  coef = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| f | result of linear model fit produced by limma functions lmFit and eBayes |
| genes | Either the name of the column in f$genes which contains the gene symbols corresponding to the gene set collection used, or a character vector with gene symbols |
| sort.by | How the gene names should be ordered: "msd" (default), "pval" or "lfc" |
| tmodFunc | The function to run the enrichment tests. Either tmodCERNOtest or tmodUtest |
| coef | If not NULL, only run tmod on these coefficients |
| ... | Further parameters passed to the tmod test function |

## Details

For each coefficient in the fit returned by the eBayes / lmFit functions from the limma package, tmodLimmaTest will order the genes run an enrichment test and return the results.

The ordering of the genes according to a certain metric is the fundament for gene enrichment analysis. tmodLimmaTest allows three orderings: p-values, "MSD" and log fold changes. The default MSD ("minimal significant difference") is the lower boundary of the 95 confidence interval for positive log fold changes, and 0 minus the upper boundary of the 95 better than ordering by p-value or by log fold change. See discussion in the package vignette.

## Value

A list with length equal to the number of coeffients. Each element is the value returned by tmod test function. The list can be directly passed to the functions tmodSummary and tmodPanelPlot.

## See Also

tmodCERNOtest, tmodUtest, tmodPlotPanel, tmodSummary

## Examples

```
## Not run:
data(Egambia)
design <- cbind(Intercept=rep(1, 30), TB=rep(c(0,1), each= 15))
if(require(limma)) {
  fit <- eBayes( lmFit(Egambia[,-c(1:3)], design))
  ret <- tmodLimmaTest(fit, genes=Egambia$GENE_SYMBOL)
  tmodSummary(ret)
  tmodPanelPlot(ret)
}

## End(Not run)
```

`tmodLimmaTopTable` *tmod's replacement for the limma topTable function*

## Description

Produce a data frame for all or for selected coefficients of a linear fit object, including log fold changes, q-values, confidence intervals and MSD.

## Usage

```
tmodLimmaTopTable(
  f,
  genelist = NULL,
  coef = NULL,
  adjust.method = "BH",
  confint = 0.95
)
```

## Arguments

| | |
|---|---|
| `f` | result of linear model fit produced by limma functions lmFit and eBayes |
| `genelist` | A data frame or a character vector with additional information on the genes / probes |
| `coef` | Which coefficients to extract |
| `adjust.method` | Which method of p-value adjustment; see "p.adjust()" |
| `confint` | Confidence interval to be calculated |

## Details

Produce a data frame for all or for selected coefficients of a linear fit object, including log fold changes, q-values, confidence intervals and MSD. For each coefficient, these four columns will be created in the output file, with the name consisting of a prefix indicating the type of the column ("msd", "logFC", "qval", "SE", "ci.L", "ci.R") and the name of the coefficient.

## Value

A data frame with all genes.

## See Also

tmodLimmaTest

## tmodPal *A selection of color palettes*

### Description

Return a preset selection of colors, adjusted by alpha

### Usage

```
tmodPal(n = NULL, set = "friendly", alpha = 0.7, func = FALSE)
```

### Arguments

| | |
|---|---|
| n | Number of colors to return (default: all for "friendly", 3 for everything else) |
| set | Which palette set (see Details). |
| alpha | 0 for maximum transparency, 1 for no transparency. |
| func | if TRUE, the returned object will be a function rather than a character vector |

### Details

A few palettes have been predefined in tmod, and this function can be used to extract them. The following palettes have been defined: * friendly – a set of distinct, colorblind-friendly colors * bwr, rwb, ckp, pkc – gradients (b-blue, r-red, w-white, c-cyan, k-blacK, p-purple) By default, either all colors are returned, or, if it is a gradient palette, only three.

### Value

Either a character vector, or, when the func parameter is TRUE, a function that takes only one argument (a single number)

## tmodPanelPlot *Plot a summary of multiple tmod analyses*

### Description

Plot a summary of multiple tmod analyses

## Usage

```
tmodPanelPlot(
  x,
  pie = NULL,
  clust = "qval",
  select = NULL,
  filter.empty.cols = FALSE,
  filter.empty.rows = TRUE,
  filter.unknown = TRUE,
  filter.rows.pval = 0.05,
  filter.rows.auc = 0.5,
  filter.by.id = NULL,
  col.labels = NULL,
  col.labels.style = "top",
  row.labels = NULL,
  row.labels.auto = "both",
  pval.thr = 10^-2,
  pval.thr.lower = 10^-6,
  plot.func = NULL,
  grid = "at",
  pie.colors = c("#0000FF", "#cccccc", "#FF0000"),
  plot.cex = 1,
  text.cex = 1,
  pie.style = "auto",
  min.e = 0.5,
  max.e = 1,
  legend.style = "auto",
  ...
)
```

## Arguments

| | |
|---|---|
| x | either a list, in which each element has been generated with a tmod test function, or the result of the tmodSummary function |
| pie | a list of data frames with information for drawing a pie chart |
| clust | whether, in the resulting data frame, the modules should be ordered by clustering them with either q-values ("qval") or the effect size ("effect"). If "sort" or NULL, the modules are sorted alphabetically by their ID. If "keep", then the order of the modules is kept. |
| select | a character vector of module IDs to show. If clust == "keep", then in that particular order. |
| filter.empty.cols | If TRUE, all elements (columns) with no enrichment below pval.thr in any row will be removed |
| filter.empty.rows | If TRUE, all modules (rows) with no enrichment below pval.thr in any column will be removed |

| | |
|---|---|
| `filter.unknown` | If TRUE, modules with no annotation will be omitted |
| `filter.rows.pval` | Rows in which no p value is below this threshold will be omitted |
| `filter.rows.auc` | Rows in which no AUC value is above this threshold will be omitted |
| `filter.by.id` | if provided, show only modules with IDs in this character vector |
| `col.labels` | Labels for the columns. If NULL, names of the elements of the list x will be used. |
| `col.labels.style` | Style of column names: "top" (default), "bottom", "both", "none" |
| `row.labels` | Labels for the modules. This must be a named vector, with module IDs as vector names. If NULL, module titles from the analyses results will be used. |
| `row.labels.auto` | Automatic generation of row labels from module data: "both" or "auto" (default, ID and title), "id" (only ID), "title" (only title), "none" (no row label) |
| `pval.thr` | Results with p-value above pval.thr will not be shown |
| `pval.thr.lower` | Results with p-value below pval.thr.lower will look identical on the plot |
| `plot.func` | Optionally, a function to be used to draw the dots. See "pvalEffectPlot" |
| `grid` | Style of a light-grey grid to be plotted; can be "none", "at" and "between" |
| `pie.colors` | character vector of length equal to the cardinality of the third dimension of the pie argument. By default: blue, grey and red. |
| `plot.cex` | a numerical value giving the amount by which the plot symbols will be magnified |
| `text.cex` | a numerical value giving the amount by which the plot text will be magnified, or a vector containing three cex values for row labels, column labels and legend, respectively |
| `pie.style` | Can be "auto" (default), "dot", "symdot", "pie", "boxpie", "rug" (see Details) |
| `min.e`, `max.e` | scale limits for the effect size (default: 0.5 and 1.0) |
| `legend.style` | Style of the legend: "auto" – automatic; "broad": pval legend side by side with effect size legend; "tall": effect size legend above pval legend |
| `...` | Any further arguments will be passed to the pvalEffectPlot function (for example, grid.color) |

## Details

This function is useful if you run an analysis for several conditions or time points and would like to summarize the information on a plot. You can use lapply() to generate a list with tmod results and use tmodPanelPlot to visualize it.

tmodPanelPlot shows a heatmap-like plot. Each row corresponds to one module, and columns correspond to the time points or conditions for which the tmod analyses were run. Each significantly enriched module is shown as a red dot. Size of the dot corresponds to the effect size (for example, AUC in the CERNO test), and intensity of the color corresponds to the q-value.

By default, tmodPanelPlot visualizes each the results of a single statistical test by a red dot, or blue and red dots if the effect sizes are both negative and positive. However, it is often interesting to

know how many of the genes in a module are significantly up- or down regulated. tmodPanelPlot can draw a pie chart based on the optional argument "pie". The argument must be a list of length equal to the length of x. Note also that the names of the pie list must be equal to the names of x. Objects returned by the function tmodDecideTests can be directly used here. The rownames of either the data frame or the array must be the module IDs.

## Value

a data frame with a line for each module encountered anywhere in the list x, two columns describing the module (ID and module title), and two columns(effect size and q value) for each element of list x.

## See Also

tmodDecideTests, tmodSummary, pvalEffectPlot, simplePie

## Examples

```
data(Egambia)
E <- Egambia[,-c(1:3)]
pca <- prcomp(t(E), scale.=TRUE)

# Calculate enrichment for first 5 PCs
gs   <- Egambia$GENE_SYMBOL
gn.f <- function(r) {
    o <- order(abs(r), decreasing=TRUE)
    tmodCERNOtest(gs[o],
                qval=0.01)
}
x <- apply(pca$rotation[,3:4], 2, gn.f)
tmodPanelPlot(x, text.cex=0.7)
```

---

# tmodPCA *PCA plot annotated with tmod*

---

## Description

Generate a PCA plot on which each dimension is annotated by a tag cloud based on tmod enrichment test.

## Usage

```
tmodPCA(
  pca,
  loadings = NULL,
  genes,
  tmodfunc = "tmodCERNOtest",
  plotfunc = pcaplot,
  mode = "simple",
```

```
  components = c(1, 2),
  plot.params = NULL,
  filter = TRUE,
  simplify = TRUE,
  legend = FALSE,
  maxn = NULL,
  plot = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| pca | Object returned by prcomp or a matrix of PCA coordinates. In the latter case, a loading matrix must be provided separately. |
| loadings | A matrix with loadings |
| genes | A character vector with gene identifiers |
| tmodfunc | Name of the tmod enrichment test function to use. Either |
| plotfunc | Function for plotting the PCA plot. See Details |
| mode | Type of the plot to generate; see Details. tmodCERNOtest or tmodUtest (tmodHGtest is not suitable) |
| components | integer vector of length two: components which components to show on the plot. Must be smaller than the number of columns in pca. |
| plot.params | A list of parameters to be passed to the plotting function. See Details |
| filter | Whether "uninteresting" modules (with no annotation) should be removed from the tag cloud |
| simplify | Whether the names of the modules should be simplified |
| legend | whether a legend should be shown |
| maxn | Maximum number of gene set enrichment terms shown on the plot (if NULL – default – all terms will be shown) |
| plot | if FALSE, no plot will be shown, but the enrichments will be calculated and returned invisibly |
| ... | Any further parameters passed to the tmod test function |

## Details

There are three types of plots that can be generated (parameter "mode"): simple, leftbottom and cross. In the "simple" mode, two enrichments are run, on on each component, sorted by absolute loadings of the PCA components. Both "leftbottom" and "cross" run two enrichment analyses on each component, one on the loadings sorted from lowest to largest, and one on the loadings sorted from largetst to lowest. Thus, two tag clouds are displayed per component. In the "leftbottom" mode, the tag clouds are displayed to the left and below the PCA plot. In the "cross" mode, the tag clouds are displayed on each of the four sides of the plot.

By default, the plotting function is plotpca. You can define your own function instead of plotpca, however, mind that in any case, there will be two parameters passed to it on the first two positions: pca and components, named "pca" and "components" respectively.

## Value

A list containing the calculated enrichments as well as the return value from the plotting function

## Examples

```
data(Egambia)
E <- as.matrix(Egambia[,-c(1:3)])
pca <- prcomp(t(E), scale.=TRUE)
group <- rep(c("CTRL", "TB"), each=15)
tmodPCA(pca,
  genes=Egambia$GENE_SYMBOL,
  components=4:3,
  plot.params=list(group=group))
```

---

# tmodSummary *Create a summary of multiple tmod analyses*

---

## Description

Create a summary of multiple tmod analyses

## Usage

```
tmodSummary(
  x,
  clust = NULL,
  filter.empty = FALSE,
  filter.unknown = TRUE,
  select = NULL,
  effect.col = NULL,
  pval.col = "adj.P.Val"
)
```

## Arguments

| | |
|---|---|
| x | list, in which each element has been generated with a tmod test function |
| clust | whether, in the resulting data frame, the modules should be ordered by clustering them with either q-values ("qval") or the effect size ("effect"). If "sort" or NULL, the modules are sorted alphabetically by their ID. If "keep", then the order of the modules is kept. |
| filter.empty | If TRUE, all elements (columns) with no significant enrichment will be removed |
| filter.unknown | If TRUE, modules with no annotation will be omitted |
| select | a character vector of module IDs to show. If clust == "keep", then in that particular order. |
| effect.col | The name of the column with the effect size (if NULL, the default, the effect size will be taken from the tmod results object attributes) |
| pval.col | The name of the p-value column |

## Details

This function is useful if you run an analysis for several conditions or time points and would like to summarize the information in a single data frame. You can use lapply() to generate a list with tmod results and use tmodSummary to convert it to a data frame.

## Value

a data frame with a line for each module encountered anywhere in the list x, two columns describing the module (ID and module title), and two columns(effect size and q value) for each element of list x.

## See Also

tmodPanelPlot

## Examples

```
## Not run:
data(Egambia)
E <- Egambia[,-c(1:3)]
pca <- prcomp(t(E), scale.=TRUE)

# Calculate enrichment for each component
gs   <- Egambia$GENE_SYMBOL
gn.f <- function(r) {
    tmodCERNOtest(gs[order(abs(r),
                decreasing=TRUE)],
                qval=0.01)
}
x <- apply(pca$rotation, 2, gn.f)
tmodSummary(x)

## End(Not run)
```

---

# tmodTagcloud *Tag cloud based on tmod results*

---

## Description

Plot a tag (word) cloud based on results from tmod enrichment.

## Usage

```
tmodTagcloud(
  results,
  filter = TRUE,
  simplify = TRUE,
  tag.col = "Title",
```

```
  min.auc = 0.5,
  max.qval = 0.05,
  plot = TRUE,
  weights.col = "auto",
  pval.col = "P.Value",
  maxn = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| results | data frame produced by one of the tmod enrichment tests |
| filter | Whether redundant and not annotated modules should be removed |
| simplify | Whether module names should be simplified |
| tag.col | Which column from results should be used as tags on the plot |
| min.auc | Minimal AUC to show (default: 0.5) |
| max.qval | Maximal adjusted p value to show (default: 0.05) |
| plot | Should the tag cloud be plotted or only returned |
| weights.col | Which column from results should be used as weights for the tag cloud |
| pval.col | Which column contains the P values which will be used to shade the tags |
| maxn | Maximum number of gene set enrichment terms shown on the plot (if NULL – default – all terms will be shown) |
| ... | Any further parameters are passed to the tagcloud function |

## Details

The tags will be generated based on results from tmod or any other suitable data frame. The data frame must contain two numeric columns, specified with "weights.col" and "pval.col", which will be used to calculate the size and shade of the tags, respectively. Furthermore, it has to contain a column with tags (parameter "tag.col", by default "Title").

Any data frame can be used as long as it contains the specified columns.

## Value

Either NULL or whatever tagcloud returns

## Examples

```
data(tmod)
fg <- getModuleMembers("LI.M127")[[1]]
bg <- tmod$gv
result <- tmodHGtest( fg, bg )
tmodTagcloud(result)
```

`tmodUtest` *Perform a statistical test of module expression*

## Description

Perform a statistical test of module expression

## Usage

```
tmodUtest(
  l,
  modules = NULL,
  qval = 0.05,
  order.by = "pval",
  filter = FALSE,
  mset = "all",
  cols = "Title",
  useR = FALSE,
  nodups = TRUE
)

tmodGeneSetTest(
  l,
  x,
  modules = NULL,
  qval = 0.05,
  order.by = "pval",
  filter = FALSE,
  mset = "all",
  cols = "Title",
  Nsim = 1000,
  nodups = TRUE
)

tmodCERNOtest(
  l,
  modules = NULL,
  qval = 0.05,
  order.by = "pval",
  filter = FALSE,
  mset = "all",
  cols = "Title",
  nodups = TRUE
)

tmodPLAGEtest(
  l,
```

```
  x,
  group,
  modules = NULL,
  qval = 0.05,
  order.by = "pval",
  mset = "all",
  cols = "Title",
  filter = FALSE,
  nodups = TRUE
)

tmodZtest(
  l,
  modules = NULL,
  qval = 0.05,
  order.by = "pval",
  filter = FALSE,
  mset = "all",
  cols = "Title",
  nodups = TRUE
)

tmodHGtest(
  fg,
  bg,
  modules = NULL,
  qval = 0.05,
  order.by = "pval",
  filter = FALSE,
  mset = "all",
  cols = "Title",
  nodups = TRUE
)
```

## Arguments

| | |
|---|---|
| l | sorted list of HGNC gene identifiers |
| modules | optional list of modules for which to make the test |
| qval | Threshold FDR value to report |
| order.by | Order by P value ("pval") or none ("none") |
| filter | Remove gene names which have no module assignments |
| mset | Which module set to use. Either a character vector ("LI", "DC" or "all", default: all) or an object of class tmod (see "Custom module definitions" below) |
| cols | Which columns from the MODULES data frame should be included in resulsts |
| useR | use the R `wilcox.test` function; slow, but with exact p-values for small samples |
| nodups | Remove duplicate gene names in l and corresponding rows from ranks |

| | |
|---|---|
| x | Expression matrix for the tmodPLAGEtest; a vector for tmodGeneSetTest |
| Nsim | for tmodGeneSetTest, number of replicates for the randomization test |
| group | group assignments for the tmodPLAGEtest |
| fg | foreground gene set for the HG test |
| bg | background gene set for the HG test |

## Details

Performs a test on either on an ordered list of genes (tmodUtest, tmodCERNOtest, tmodZtest) or on two groups of genes (tmodHGtest). tmodUtest is a U test on ranks of genes that are contained in a module.

tmodCERNOtest is also a nonparametric test working on gene ranks, but it originates from Fisher's combined probability test. This test weights genes with lower ranks more, the resulting p-values better correspond to the observed effect size. In effect, modules with small effect but many genes get higher p-values than in case of the U-test.

tmodPLAGEtest is based on the PLAGE, "Pathway level analysis of gene expression" published by Tomfohr, Lu and Kepler (2005), doi 10.1186/1471-2105-6-225. In essence it is just a t-test run on module eigengenes, but it performs really well. This approach can be used with any complex linear model; for this, use the function eigengene(). See users guide for details.

tmodZtest works very much like tmodCERNOtest, but instead of combining the rank-derived p-values using Fisher's method, it uses the Stouffer method (known also as the Z-transform test).

tmodGeneSetTest is an implementation of the function geneSetTest from the limma package (note that tmodUtest is equivalent to the limma's wilcoxGST function).

For a discussion of the above three methods, read M. C. Whitlock, "Combining probability from independent tests: the weighted Z-method is superior to Fisher's approach", J. Evol. Biol. 2005 (doi: 10.1111/j.1420-9101.2005.00917.x) for further details.

tmodHGtest is simply a hypergeometric test.

In tmod, two module sets can be used, "LI" (from Li et al. 2013), or "DC" (from Chaussabel et al. 2008). Using the parameter "mset", the module set can be selected, or, if mset is "all", both of sets are used.

## Value

The statistical tests return a data frame with module names, additional statistic (e.g. enrichment or AUC, depending on the test), P value and FDR q-value (P value corrected for multiple testing using the p.adjust function and Benjamini-Hochberg correction. The data frame has class 'colorDF' (see package colorDF for details), but except for printing using colors on the terminal behaves just like an ordinary data.frame. To strip the coloring, use [colorDF::uncolor()].

## Custom module definitions

Custom and arbitrary module, gene set or pathway definitions can be also provided through the mset option, if the parameter is a list rather than a character vector. The list parameter to mset must contain the following members: "MODULES", "MODULES2GENES" and "GENES".

"MODULES" and "GENES" are data frames. It is required that MODULES contains the following columns: "ID", specifying a unique identifier of a module, and "Title", containing the description of the module. The data frame "GENES" must contain the column "ID".

The list MODULES2GENES is a mapping between modules and genes. The names of the list must correspond to the ID column of the MODULES data frame. The members of the list are character vectors, and the values of these vectors must correspond to the ID column of the GENES data frame.

## See Also

tmod-package

## Examples

```
data(tmod)
fg <- tmod$MODULES2GENES[["LI.M127"]]
bg <- tmod$GENES$ID
result <- tmodHGtest( fg, bg )

## A more sophisticated example
## Gene set enrichment in TB patients compared to
## healthy controls (Egambia data set)
## Not run:
data(Egambia)
library(limma)
design <- cbind(Intercept=rep(1, 30), TB=rep(c(0,1), each= 15))
fit <- eBayes( lmFit(Egambia[,-c(1:3)], design))
tt <- topTable(fit, coef=2, number=Inf, genelist=Egambia[,1:3] )
tmodUtest(tt$GENE_SYMBOL)
tmodCERNOtest(tt$GENE_SYMBOL)

## End(Not run)
```

---

# tmod_ids *Query and set IDs of gene sets in a tmodGS object*

---

## Description

Query and set IDs (tmod_id) or Titles (tmod_title) of gene sets in a tmodGS object

## Usage

```
tmod_ids(x)

tmod_ids(x) <- value

tmod_titles(x)

tmod_titles(x) <- value
```

### Arguments

| | |
|---|---|
| x | an object of class tmodGS |
| value | a character vector of unique IDs |

### Value

Returns character vector corresponding to x$gs$ID

### Examples

```
data(tmod)
mset <- tmod[ c("LI.M37.0", "LI.M75", "LI.M3") ]
tmod_ids(mset)
tmod_ids(mset) <- c("em", "pstrem", "bzdrem")
tmod_titles(mset) <- c("foo", "bar", "baz")
mset$gs
```

---

## upset *Upset plot*

---

### Description

Upset plots help to interpret the results of gene set enrichment.

### Usage

```
upset(
  modules,
  mset = NULL,
  min.size = 2,
  min.overlap = 2,
  max.comb = 4,
  min.group = 2,
  value = "number",
  cutoff = NULL,
  labels = NULL,
  group.stat = "jaccard",
  group.cutoff = 0.1,
  group = TRUE,
  pal = brewer.pal(8, "Dark2"),
  lab.cex = 1
)
```

## Arguments

| | |
|---|---|
| modules | optional list of modules for which to make the test |
| mset | Which module set to use. Either a character vector ("LI", "DC" or "all", default: all) or an object of class tmod (see "Custom module definitions" below) |
| min.size | minimal number of modules in a comparison to show |
| min.overlap | smallest overlap (number of elements) between two modules to plot |
| max.comb | Maximum number of combinations to show (i.e., number of dots on every vertical segment in the upset plot) |
| min.group | Minimum number of modules in a group. Group with a smaller number of members will be ignored. Change this value to 1 to see also modules which could not be grouped. |
| value | what to show on the plot: "number" (number of common elements; default), "soerensen" (Sørensen–Dice coefficient), "overlap" (Szymkiewicz–Simpson coefficient) or "jaccard" (Jaccard index) |
| cutoff | Combinations with the ‘value‘ below cutoff will not be shown. |
| labels | Labels for the modules. Character vector with the same length as ‘modules‘ |
| group.stat | Statistics for finding groups (can be "number", "overlap", "soerensen" or "jaccard"; see function modOverlaps) |
| group.cutoff | cutoff for group statistics |
| group | Should the modules be grouped by the overlap? |
| pal | Color palette to show the groups. |
| lab.cex | Initial cex (font size) for labels |

## Details

The plot consists of three parts. The main part shows the overlaps between the different modules (module can be a gene set, for example). Each row corresponds to one module. Each column corresponds to an intersection of one or more gene sets. Dots show which gene sets are in that combination. Which combinations are shown depends on the parameters ‘min.overlap‘ (which is the cutoff for the similarity measure specified by the ‘value‘ parameter), the parameter ‘min.group‘ which specifies the minimum number of modules in a group and the parameter ‘max.comb‘ which specifies the maximum number of combinations tested (too many combinations are messing the plot).

Above the intersections, you see a plot showing a similarity measure of the intersected gene sets. By default it is the number of module members (genes in case of a gene set), but several other measures (e.g. the Jaccard index) are also implemented.

To the left are the module descriptions (parameter ‘label‘; if label is empty, the labels are taken from the mset object provided or, if that is NULL, from the default tmod module set). The function attempts to scale the text in such a way that all labels are visible.

By default, upset attempts to group the modules. This is done by defining a similarity measure (by default the Jaccard index, parameter ‘group.stat‘) and a cutoff threshold (parameter ‘group.cutoff‘).

## Value

upset returns invisibly the identified module groups: a list of character vectors.

## See Also

[modGroups()], [modOverlaps()]

## Examples

```
## Not run:
data(Egambia)
design <- cbind(Intercept=rep(1, 30), TB=rep(c(0,1), each= 15))
library(limma)
fit <- eBayes( lmFit(Egambia[,-c(1:3)], design))
tt <- topTable(fit, coef=2, number=Inf, genelist=Egambia[,1:3] )
res <- tmodCERNOtest(tt$GENE_SYMBOL)

upset(res$ID, group.cutoff=.1, value="jaccard")

## End(Not run)
```

---

# vaccination *Transcriptomic responses to vaccination*

---

## Description

Transcriptomic responses to vaccination

## Format

Data frame with one row per gene containing log fold changes and FDR (q values) for the Fluad vaccine as compared to placebo on day 0, day 1, day 2 and day 3 after the vaccination.

## Details

The data shows the time course of transcriptomic responses to influenza vaccination in healthy volunteers. The source of the data is GEO project PRJNA515032, associated with the following paper:

Weiner, January, et al. "Characterization of potential biomarkers of reactogenicity of licensed antiviral vaccines: randomized controlled clinical trials conducted by the BIOVACSAFE consortium." Scientific reports 9.1 (2019): 1-14.

For the data set, 3000 genes with top variance were chosen.

# Index