Package 'GOsummaries'

April 14, 2017

Version 2.8.0

Date 2016-03-16

License GPL (>= 2)

Description A package to visualise Gene Ontology (GO) enrichment analysis results on gene lists arising from different analyses such clustering or PCA. The significant GO categories are visualised as word clouds that can be combined with different plots summarising the underlying data.

Title Word cloud summaries of GO enrichment analysis

Author Raivo Kolde <raivo.kolde@eesti.ee>

Maintainer Raivo Kolde <raivo.kolde@eesti.ee>

Depends R (>= 2.15), Rcpp

Imports plyr, grid, gProfileR, reshape2, limma, ggplot2, gtable

Suggests vegan

LinkingTo Rcpp

URL https://github.com/raivokolde/GOsummaries

biocViews GeneExpression, Clustering, GO, Visualization

RoxygenNote 5.0.1

NeedsCompilation yes

R topics documented:

`GOsummaries-package` *Word cloud summaries of GO enrichment analysis*

Description

A package to visualize Gene Ontology (GO) enrichment analysis results on gene lists arising from different analyses such clustering or PCA. The significant GO categories are visualised as word clouds that can be combined with different plots summarizing the underlying data.

Details

The goal of GOsummaries package is to draw figures that can be used in presentations and articles. To draw them, the user should first construct a `gosummaries` object and then use its plot function on it. One can start constructing the `gosummaries` object from gene lists, with filling in all the necessary information step by step. However, there are some convenience functions for different classes of common analysis results. See `gosummaries.kmeans`, `gosummaries.MArrayLM` and `gosummaries.prcomp` corresponding to k-means, limma and PCA results.

The `plot.gosummaries` describes how to customize the plots.

The word cloud drawing function `plotWordcloud` in this package is implemented largely based on the code from package `wordcloud`, but with slight tweaks: it uses `grid` graphics, has some additional layout options, has more intelligent options to scale the text sizes to fit the picture and, finally, should be a bit faster since larger part of the algorithm was implemeted in C++.

`add_expression.gosummaries`
Add expression data to gosummaries object

Description

Function to add expression data and its annotations to a gosummaries object.

Usage

```
add_expression.gosummaries(gosummaries, exp, annotation = NULL)
```

Arguments

`gosummaries`	a gosummaries object
`exp`	an expression matrix, with row names corresponding to the names in the Gene_lists slot
`annotation`	a `data.frame` describing the samples, its row names should match with column names of `exp`

Details

The data is added to the object in a "long" format so it would be directly usable by the ggplot2 based panel drawing functions panel_boxplot etc. For each component it produces a data frame with columns:

- **x** : sample IDs for the x axis, their factor order is the same as on the columns of input matrix
- **y** : expression values from the matrix
- . . . : sample annotation columns from the annotation table that can be displayed on figure as colors.

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Examples

```
## Not run:
data(gs_limma)
data(tissue_example)

# Add just expression without annotations
gs_limma_exp1 = add_expression.gosummaries(gs_limma, exp =
tissue_example$exp)

print(gs_limma_exp1)

# Add expression with annotations
gs_limma_exp2 = add_expression.gosummaries(gs_limma, exp =
tissue_example$exp, annotation = tissue_example$annot)

print(gs_limma_exp1)

## End(Not run)
```

---

customize *Customization function for panel*

Description

This function is supposed to make small changes in the panel function appearance like changing color scheme for example. It has to match with the output of the corresponding panel function. Check examples in plot.gosummaries to see how to write one yourself.

Usage

```
customize(p, par)
```

Arguments

p	a ggplot2 plot object
par	parameters object like in panel_boxplot

Value

a ggplot2 plot object with added customizations

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Examples

```
## Not run:
data(gs_limma_exp)

cust = function(p, par){
    p = p + scale_fill_brewer(par$classes, type = "qual", palette = 1)
    return(p)
}

plot(gs_limma_exp, classes = "Tissue", panel_plot = panel_boxplot,
        panel_customize = cust, fontsize = 8)

## End(Not run)
```

---

gosummaries *Constructor for gosummaries object*

Description

Constructor for gosummaries object that contains all the necessary information to draw the figure, like gene lists and their annotations, expression data and all the relevant texts.

Usage

```
gosummaries(x = NULL, ...)

## Default S3 method:
gosummaries(x = NULL, wc_data = NULL,
  organism = "hsapiens", go_branches = c("BP", "ke", "re"),
  max_p_value = 0.01, min_set_size = 50, max_set_size = 1000,
  max_signif = 40, ordered_query = TRUE, hier_filtering = "moderate",
  score_type = "p-value", wc_algorithm = "middle",
  wordcloud_legend_title = NULL, ...)
```

Arguments

x	list of arrays of gene names (or list of lists of arrays of gene names)
wc_data	precalculated GO enrichment results (see Details)
organism	the organism that the gene lists correspond to. The format should be as follows: "hsapiens", "mmusculus", "scerevisiae", etc.
go_branches	GO tree branches and pathway databases as denoted in g:Profiler (Possible values: BP, CC, MF, ke, re)

max_p_value	threshold for p-values that have been corrected for multiple testing
min_set_size	minimal size of functional category to be considered
max_set_size	maximal size of functional category to be considered
max_signif	maximal number of categories returned per query
ordered_query	logical showing if the lists are ordered or not (it determines if the ordered query algorithm is used in g:Profiler)
hier_filtering	a type of hierarchical filtering used when reducing the number of g:Profiler results (see gprofiler for further information)
score_type	indicates the type of scores in wc_data. Possible values: "p-value" and "count"
wc_algorithm	the type of wordcloud algorithm used. Possible values are "top" that puts first word to the top corner and "middle" that puts first word to the middle.
wordcloud_legend_title	title of the word cloud legend, should reflect the nature of the score
...	additional parameters for gprofiler function

Details

The object is a list of "components", with each component defined by a gene list or a pair of gene lists. Each "component" has the slots as follows:

- **Title**: title string of the component. (Default: the names of the gene lists)
- **Gene_lists**: list of one or two gene lists
- **WCD**: g:Profiler results based on the Gene_lists slot or user entered table.
- **Data**: the related data (expression values, PCA rotation, ...) that is used to draw the "panel" i.e. theplot above the wordclouds. In principle there is no limitation what kind of data is there as far as the function that is provided to draw that in plot.gosummaries can use it.
- **Percentage**: a text that is drawn on the right top corner of every component. In case of PCA this is the percentage of variation the component explains, by default it just depicts the number of genes in the Gene_lists slot.

Some visual parameters are stored in the attributes of gosummaries object: score_type tells how to handle the scores associated to wordclouds, wc_algorithm specifies the wordcloud layout algorithm and wordcloud_legend_title specifies the title of the wordcloud. One can change them using the attr function.

The word clouds are specified as data.frames with two columns: "Term" and "Score". If one wants to use custom data for wordclouds, instead of the default GO enrichment results, then this is possible to specify parameter wc_data. The input structure is similar to the gene list input, only instead of gene lists one has the two column data.frames.

The GO enrichment analysis is performed using g:Profiler web toolkit and its associated R package gProfileR. This means the computer has to have internet access to annotate the gene lists. Since g:Profiler can accept a wide range of gene IDs then user usually does not have to worry about converitng the gene IDs into right format. To be absolutely sure the tool recognizes the gene IDs one can check if they will give any results in http://biit.cs.ut.ee/gprofiler/gconvert.cgi.

There can be a lot of results for a typical GO enrichment analysis but usually these tend to be pretty redundant. Since one can fit only a small number of categories into a word cloud we have to bring down the number of categories to show an reduce the redundancy. For this we use hierarchical filtering option \"moderate\" in g:Profiler. In g:Profiler the categories are grouped together when they share one or more enriched parents. The \"moderate\" option selects the most significant category from each of such groups. (See more at http://biit.cs.ut.ee/gprofiler/)

The slots of the object can be filled with custom information using a function add_to_slot.gosummaries.

By default the Data slot is filled with a dataset that contains the number of genes in the Gene_lists slot. Expression data can be added to the object for example by using function add_expression.gosummaries. It is possible to derive your own format for the Data slot as well, as long as a panel plotting function for this data is alaso provided (See panel_boxplot for further information).

There are several constructors of gosummaries object that work on common analysis result objects, such as gosummaries.kmeans, gosummaries.MArrayLM and gosummaries.prcomp corresponding to k-means, limma and PCA results.

Value

A gosummaries type of object

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Raivo Kolde <raivo.kolde@eesti.ee>

See Also

gosummaries.kmeans, gosummaries.MArrayLM, gosummaries.prcomp

Examples

```
## Not run:
# Define gene lists
genes1 = c("203485_at", "209469_at", "209470_s_at", "203999_at",
"205358_at", "203130_s_at", "210222_s_at", "202508_s_at", "203001_s_at",
"207957_s_at", "203540_at", "203000_at", "219619_at", "221805_at",
"214046_at", "213135_at", "203889_at", "209990_s_at", "210016_at",
"202507_s_at", "209839_at", "204953_at", "209167_at", "209685_s_at",
"211276_at", "202391_at", "205591_at",
"201313_at")
genes2 = c("201890_at", "202503_s_at", "204170_s_at", "201291_s_at",
"202589_at", "218499_at", "209773_s_at", "204026_s_at", "216237_s_at",
"202546_at", "218883_s_at", "204285_s_at", "208659_at", "201292_at",
"201664_at")


gl1 = list(List1 = genes1,  List2 = genes2) # One list per component
gl2 = list(List = list(genes1, genes2)) # Two lists per component

# Construct gosummaries objects
gs1 = gosummaries(gl1)
gs2 = gosummaries(gl2)

plot(gs1, fontsize = 8)
plot(gs2, fontsize = 8)

# Changing slot contents using using addToSlot.gosummaries
gs1 = add_to_slot.gosummaries(gs1, "Title", list("Neurons", "Cell lines"))

# Adding expression data
data(tissue_example)
```

```
gs1 = add_expression.gosummaries(gs1, exp = tissue_example$exp, annotation =
tissue_example$annot)
gs2 = add_expression.gosummaries(gs2, exp = tissue_example$exp, annotation =
tissue_example$annot)

plot(gs1, panel_par = list(classes = "Tissue"), fontsize = 8)
plot(gs2, panel_par = list(classes = "Tissue"), fontsize = 8)

## End(Not run)

# Using custom annotations for word clouds
wcd1 = data.frame(Term = c("KLF1", "KLF2", "POU5F1"), Score = c(0.05, 0.001,
0.0001))
wcd2 = data.frame(Term = c("CD8", "CD248", "CCL5"), Score = c(0.02, 0.005,
0.00001))

gs = gosummaries(wc_data = list(Results1 = wcd1, Results2 = wcd2))
plot(gs)

gs = gosummaries(wc_data = list(Results = list(wcd1, wcd2)))
plot(gs)

# Adjust wordcloud legend title
gs = gosummaries(wc_data = list(Results = list(wcd1, wcd2)),
wordcloud_legend_title = "Significance score")
plot(gs)
```

`gosummaries.kmeans` *Prepare gosummaries object based on k-means results*

Description

The gosummaries object is created based on the genes in the clusters, it is possible to add corresponding gene expression data as well.

Usage

```
## S3 method for class 'kmeans'
gosummaries(x, exp = NULL, annotation = NULL,
  components = 1:length(x$size), organism = "hsapiens", ...)
```

Arguments

`x`	an object of class `kmeans`
`exp`	an expression matrix, with row names corresponding to the names of the genes in clusters (Optional)
`annotation`	a `data.frame` describing the samples, its row names should match with column names of `exp` (Optional)
`components`	numeric vector of clusters to annotate
`organism`	the organism that the gene lists correspond to. The format should be as follows: "hsapiens", "mmusculus", "scerevisiae", etc.
`...`	GO annotation filtering parameters as defined in `gosummaries.default`

Details

The k-means clustering of expression matrix naturally defines a set of gene lists that can be annotated functionally and displayed as a GOsummaries figure. This functon takes in a `kmeans` object and and converts it to a `gosummaries` object that can be plotted. If expression matrix is attached then the panel shows the expression values for each gene as boxplots, if not then number of genes is displayed

It is advisable to filter some genes out before doing the clustering since the very large gene lists (more than 2000 genes) might fail the annotation step and are usually not too specific either.

Value

A gosummaries object.

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Examples

```
## Not run:
data(tissue_example)

# Filter genes and perform k-means
sd = apply(tissue_example$exp, 1, sd)
exp2 = tissue_example$exp[sd > 0.75,]
exp2 = exp2 - apply(exp2, 1, mean)
kmr = kmeans(exp2, centers = 6, iter.max = 100)

# Create gosummaries object
gs_kmeans = gosummaries(kmr, exp = exp2, annotation = tissue_example$annot)
plot(gs_kmeans, panel_height = 0, components = 1:3, fontsize = 8)
plot(gs_kmeans, classes = "Tissue", components = 1:3, fontsize = 8)

## End(Not run)
```

---

gosummaries.MArrayLM *Prepare gosummaries object based on limma results*

Description

The gosummaries object is created based on the differentially expresed genes, each contrast defines one component.

Usage

```
## S3 method for class 'MArrayLM'
gosummaries(x, p.value = 0.05, lfc = 1,
  adjust.method = "fdr", exp = NULL, annotation = NULL,
  components = 1:ncol(x), show_genes = FALSE, gconvert_target = "NAME",
  n_genes = 30, organism = "hsapiens", ...)
```

Arguments

x	an object of class MArrayLM
p.value	p-value threshold as defined in topTable
lfc	log fold change threshold as defined in topTable
adjust.method	multiple testing adjustment method as defined in topTable
exp	an expression matrix, with row names corresponding to the names of the genes in clusters (Optional)
annotation	a data.frame describing the samples, its row names should match with column names of exp (Optional)
components	numeric vector of comparisons to annotate
show_genes	logical showing if GO categories or actual genes are shown in word clouds
gconvert_target	specifies gene ID format for genes showed in word cloud. The name of the format is passed to gconvert, if NULL original IDs are shown.
n_genes	maximum number of genes shown in a word cloud
organism	the organism that the gene lists correspond to. The format should be as follows: "hsapiens", "mmusculus", "scerevisiae", etc.
...	GO annotation filtering parameters as defined in gosummaries.default

Details

The usual differential expression analysis involves making several comparisons between treatments ehere each one yields an up and down regulated gene list. In a GOsummaries figure each comparison is displayed as one component with two wordclouds. If expression matrix is attached then the panel shows the expression values for each gene as boxplots, if not then number of genes is displayed

It is possible to show the gene names instead of GO annotations in the wordclouds. The word sizes in wordclouds are defined by the limma p-values. As the gene identifiers in expression matrices are usually rather unintelligible then they are automatically converted into gene names using gconvert function. It is possible to show also the original identifiers by setting gconvert_target to NULL. This can be useful if the values do not correspond to genes, but for example metabolites.

Value

A gosummaries object.

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Examples

```
## Not run:
data(tissue_example)

# Do the t-test comparisons
mm = model.matrix(~ factor(tissue_example$annot$Tissue) - 1)
colnames(mm) = make.names(levels(factor(tissue_example$annot$Tissue)))

contrast = limma::makeContrasts(brain - cell.line,
```

```
                                hematopoietic.system - muscle,
                                cell.line - hematopoietic.system,
                                levels = colnames(mm))

fit = limma::lmFit(tissue_example$exp, mm)
fit = limma::contrasts.fit(fit, contrast)
fit = limma::eBayes(fit)

gs_limma = gosummaries(fit)
gs_limma_exp = gosummaries(fit, exp = tissue_example$exp,
                           annotation = tissue_example$annot)

plot(gs_limma, fontsize = 8)
plot(gs_limma, panel_height = 0, fontsize = 8)
plot(gs_limma_exp, classes = "Tissue", fontsize = 8)

## End(Not run)
```

`gosummaries.matrix` *Prepare gosummaries object based on Multi Dimensional Scaling (MDS) results*

Description

The Multi Dimensional Scaling (MDS) results are converted into a gosummaries object, by finding genes that have most significant Spearman correlations with each component.

Usage

```
## S3 method for class 'matrix'
gosummaries(x, exp = NULL, annotation = NULL,
  components = 1:min(ncol(x), 10), show_genes = FALSE,
  gconvert_target = "NAME", n_genes = ifelse(show_genes, 30, 500),
  organism = "hsapiens", ...)
```

Arguments

`x`	a matrix representation of multi dimensional scaling result, rows correspond to samples
`exp`	an expression matrix, with columns corresponding to samples (these have to be in the same order as in `x`)
`annotation`	a `data.frame` describing the samples, its row names should match with column names of `exp` (Optional)
`components`	numeric vector of comparisons to annotate
`show_genes`	logical showing if GO categories or actual genes are shown in word clouds
`gconvert_target`	specifies gene ID format for genes showed in word cloud. The name of the format is passed to `gconvert`, if NULL original IDs are shown.
`n_genes`	maximum number of genes shown in a word cloud
`organism`	the organism that the gene lists correspond to. The format should be as follows: "hsapiens", "mmusculus", "scerevisiae", etc.
`...`	GO annotation filtering parameters as defined in `gosummaries.default`

Details

This visualisation of MDS results is very similar to the one performed by gosummaries.prcomp. Difference from PCA is that, in general, we do not have the loadings for individual genes that could be used to associate genes with components. However, it is possible to find genes that are most correlated with each component. This function uses Spearman correlation coefficient to find most correlated features. The significance of the correlation values is decided using he approximation with t-distribution.

The function can also display genes instead of their GO annotations, while the sizes of the gene names correspond to the Spearman correlation p-values. The corresponding parameters are described in more detail in gosummaries.MArrayLM. This feature is important in applications, like metabolomics and metagenomics, where the features are not genes and it is not possible to run GO enrichment analysis.

Value

A gosummaries object.

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Examples

```
## Not run:
library(vegan)

data("metagenomic_example")

# Run Principal Coordinate Analysis on Bray-Curtis dissimilarity matrix
pcoa = cmdscale(vegdist(t(metagenomic_example$otu), "bray"), k = 3)

# By turning off the GO analysis we can show the names of taxa
gs = gosummaries(pcoa, metagenomic_example$otu, metagenomic_example$annot,
                 show_genes = TRUE, gconvert_target = NULL, n_genes = 30)

plot(gs, class = "BodySite", fontsize = 8)

## End(Not run)
```

---

gosummaries.prcomp *Prepare gosummaries object based on PCA results*

Description

The PCA results are converted into a gosummaries object, by extracting genes with the largest positive and negative weights from each component.

Usage

```
## S3 method for class 'prcomp'
gosummaries(x, annotation = NULL, components = 1:10,
  show_genes = FALSE, gconvert_target = "NAME",
  n_genes = ifelse(show_genes, 30, 500), organism = "hsapiens", ...)
```

Arguments

x	an object of class prcomp
annotation	a data.frame describing the samples, its row names should match with column names of the projection matrix in x
components	numeric vector of components to include
show_genes	logical showing if GO categories or actual genes are shown in word clouds
gconvert_target	specifies gene ID format for genes showed in word cloud. The name of the format is passed to gconvert, if NULL original IDs are shown.
n_genes	shows the number of genes used for annotating the component, in case gene names are shown, it is the maximum number of genes shown in a word cloud
organism	the organism that the gene lists correspond to. The format should be as follows: "hsapiens", "mmusculus", "scerevisiae", etc
...	GO annotation filtering parameters as defined in gosummaries.default

Details

The usual visualisation of PCA results displays the projections of sample expression on the principal axes. It shows if and how the samples cluster, but not why do they behave like that. Actually, it is possible to go further and annotate the axes by studying genes that have the largest influence in the linear combinations that define the principal components. For example, high expression of genes with large negative weights pushes the samples projection to the negative side of the principal axis and large positive weigths to the positive side. If a sample has highly expressed genes in both groups it stays most probably in the middle. If we annotate functionally the genes with highest positive and negative weights for each of the principal axes, then it is possible to say which biological processes drive the separation of samples on them.

This function creates a gosummaries object for such analysis. It expects the results of prcomp function. It assumes that the PCA was done on samples and, thus, the row names of the rotation matrix can be interpreted as gene names. For each component it annotates n_genes elements with highest positive and negative weights.

The function can also display genes instead of their GO annotations, while the sizes of the gene names correspond to the PCA loadings. The corresponding parameters are described in more detail in gosummaries.MArrayLM.

Value

A gosummaries object.

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Examples

```
## Not run:
data(tissue_example)

pcr = prcomp(t(tissue_example$exp))
gs_pca = gosummaries(pcr, annotation = tissue_example$annot)

plot(gs_pca, classes = "Tissue", components = 1:3, fontsize = 8)
```

```
## End(Not run)

# Read metabolomic data
data(metabolomic_example)

pca = prcomp(t(metabolomic_example$data))

# Turn off GO enricment, since it does not work on metabolites
gs = gosummaries(pca, annotation = metabolomic_example$annot,
                 show_gene = TRUE, gconvert_target = NULL)
plot(gs, class = "Tissue", components = 1:3, fontsize = 8)
```

`is.gosummaries` *Functions for working with gosummaries object*

Description

Functions for working with gosummaries object

Usage

```
is.gosummaries(x)

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

## S3 method for class 'gosummaries'
x[i, ...]

add_to_slot.gosummaries(x, slot, values)
```

Arguments

`x`	a gosummaries object
`i`	index
`slot`	the component slot name to be filled (e.g Title, Percentage, etc.)
`values`	list of values where each element corresponds to one component
`gosummaries`	a gosummaries object
`...`	not used

Details

Method [ensures that subsetting gosummaries object will not lose the attributes.

`add_to_slot.gosummaries` allows to add values to specific slots in the gosummaries object

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Examples

```
data(gs_kmeans)

# Add new title to the components
gs_kmeans_new = add_to_slot.gosummaries(gs_kmeans, "Title",
as.list(paste("K-means cluster", 1:length(gs_kmeans))))

print(gs_kmeans_new)
```

`metabolomic_example` *Example metabolomic dataset*

Description

`metabolomic_example` is a dataset extracted from York *et al* (Metabolights ID: MTBLS30), it contains a subset of 120 wild-type samples from 4 tissues: heart, skeletal muscle, liver and brain.

Details

The dataset is a list with 2 slots

1. `data` - metabolite concentration matrix;
2. `annot` - annotation data frame.

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

References

York, B., Sagen, J. V., Tsimelzon, A., Louet, J. F., Chopra, A. R., Reineke, E. L., et al. (2013). Research resource: tissue- and pathway-specific metabolomic profiles of the steroid receptor coactivator (SRC) family. Mol Endocrinol, 27(2), 366-380.

`metagenomic_example` *Example metagenomic dataset*

Description

`metagenomic_example` is a small sample of Human Microbiome Project 16S dataset for finding biomarkers characterizing different level of oxygen availability in different bodysites. This was downloaded from http://huttenhower.sph.harvard.edu/webfm_send/129.

Details

The dataset is a list with 2 slots

1. `otu` - otu table for this experiment;
2. `annot` - annotation data frame.

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

`panel_boxplot` *Panel drawing functions*

Description

These functions are used to draw the panel portion of every component based on the Data slots in gosummaries object. These concrete functions assume the data is presented as done by `add_expression.gosummaries`. They provide three options: boxplot, violin plot (which shows the distrubution more precisely) and both combined. Additionally it is possible to combine the values from one each functional group into one box/violin plot using the corresponding functions ending with "_classes".

Usage

```
panel_boxplot(data, fontsize = 10, par)

panel_violin(data, fontsize = 10, par)

panel_violin_box(data, fontsize = 10, par)

panel_boxplot_classes(data, fontsize = 10, par)

panel_violin_classes(data, fontsize = 10, par)

panel_violin_box_classes(data, fontsize = 10, par)
```

Arguments

`data`	the data from Data slot of the gosummaries object
`fontsize`	fontsize in points, mainly used to ensure that the legend fontsizes match
`par`	additional parameters for drawing the plot, given as list. These functions use only `classes` slot for figuring out which parameter to use for coloring the "geom"-s. However, when building a custom function it provides a way to give extra parameters to the plotting function.

Details

These functions specify in principle the general setting for the panels, like which "geom"-s, how the data is transformed and summarized, etc. To make small adjustments to the figure such as changing color scheme, write your own customization function (See `customize` as example).

It is possible to write your own panel plotting function, as long as the parameters used and the return value are similar to what is specified here. When writing a new panel function one only has to make sure that it matches the data given in the Data slot of the gosummaries object.

Value

It returns a function that can draw a ggplot2 plot of the data in Data slot of a gosummaries object. The legend and the actual plots for the panels are extracted later from the figure produced by this function.

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Examples

```
## Not run:
data(gs_kmeans)

# Draw default version with plot_boxplot
plot(gs_kmeans, components = 1:3, classes = "Tissue")

# Define alternative where one boxplot summarises all values in one class
plot_classes = function(data, fontsize, par){
    qplot(x = data[, par$classes], y = data$y, geom = "boxplot",
          fill = data[, par$classes]) + theme_bw()
}

plot(gs_kmeans, components = 1:3, panel_plot = plot_classes, classes = "Tissue")


# Flip the boxplots to make them more comparable
plot_classes = function(data, fontsize, par){
    qplot(x = data[, par$classes], y = data$y, geom = "boxplot",
        fill = data[, par$classes]) + coord_flip() + theme_bw()
}

plot(gs_kmeans, components = 1:3, panel_plot = plot_classes, classes = "Tissue")

## End(Not run)
```

---

plot.gosummaries *Plot the GOsummaries figure*

Description

The function to draw a GOsummaries figure based on a `gosummaries` object. The GOsummaries figure consists of several components each defined by a gene list ora a pair of them. The GO annotations of them are shown as wordclouds. Optionally one can draw related (expression) data on panels atop of the wordclouds.

Usage

```
## S3 method for class 'gosummaries'
plot(x, components = 1:min(10, length(x)),
  classes = NA, panel_plot = NULL, panel_customize = NULL,
  panel_par = list(), panel_height = 5, panel_width = 30, fontsize = 10,
  term_length = 35, wordcloud_colors = c("grey70", "grey10"),
  wordcloud_legend_title = NULL, filename = NA, ...)
```

Arguments

x	a gosummaries object
components	index for the components to draw.
classes	name of the variable from annotation data.frame that defines the colors in the plot
panel_plot	plotting function for panel
panel_customize	customization function for the panel plot, menat for making small changes like changing colour scheme
panel_par	list of arguments passed on to panel_plot function
panel_height	panel height as number of lines, with given fontsize. If set to 0 no panel is drawn.
panel_width	panel width in lines of text
fontsize	font size used throughout the figure in points
term_length	maximum length of the dispalyed GO categories in characters, longer names are cropped to this size
wordcloud_colors	two element vector of colors to define color scheme for displaying the enrichment p-values across the wordclouds. First element defines the color for category with worst p-value and the second for the word with the best. Set the same value for both if you want to remove the color scale and the legend.
wordcloud_legend_title	title of wordcloud legend
filename	file path where to save the picture. Filetype is decided by the extension in the path. Currently following formats are supported: png, pdf, tiff, bmp, jpeg. Even if the plot does not fit into the plotting window, the file size is calculated so that the plot would fit there.
...	not used

Details

In most cases the function can decide which type of plot to draw into the panel part. If there is no data explicitly put into the Data slots of the gosummaries object, it just draws a horizontal barplot with the numbers of genes. On visualizing the PCA data it draws histogram of the samples on the principal axes. For clustering and differential expression it draws the boxplot of expression values.

Value

The gtable object containing the figure

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Examples

```
## Not run:
data(gs_limma)

# Default plot
plot(gs_limma, fontsize = 8)

# Omitting the panel area
plot(gs_limma, panel_height = 0, fontsize = 8)

# Selecting only certain components
plot(gs_limma, components = c(1, 3), fontsize = 8)

# Cutting the longer terms shorter (see right wordcloud on first component)
plot(gs_limma, term_length = 20, fontsize = 8)

# Change wordcloud colors
plot(gs_limma, term_length = 20, wordcloud_colors = c("#C6DBEF", "#08306B"),
     fontsize = 8)

# Adjust panel plot type (see panel_boxplot help for options)
data(gs_kmeans)

plot(gs_kmeans, panel_plot = panel_violin, classes = "Tissue", components =
     1:2, fontsize = 8)
plot(gs_kmeans, panel_plot = panel_violin_box, classes = "Tissue",
     components = 1:2, fontsize = 8)

# Adjust colorscheme for plot (see customize help for more information)
cust = function(p, par){
  p = p + scale_fill_brewer(par$classes, type = "qual", palette = 2)
  return(p)
}
plot(gs_kmeans, panel_plot = panel_violin, panel_customize = cust,
     classes = "Tissue", components = 1:2, fontsize = 8)

## End(Not run)
```

---

plotWordcloud *Plot a wordcloud given words and frequencies*

Description

General grid based wordcloud drawing function

Usage

```
plotWordcloud(words, freq, rot.per = 0.3, max_min = c(1, 0.1),
  scale = 0.4, min.freq = 3, max.words = Inf, random.order = FALSE,
  colors = "black", random.colors = FALSE, fontface = 1,
  algorithm = "circle", tryfit = TRUE, add = FALSE, grob = FALSE,
  dimensions = unit(c(1, 1), "npc"))
```

Arguments

words	vector of words to draw
freq	frequencies for words, has to be the same length as words vector
rot.per	percentage of vertical words
max_min	relative scales to adjust the size difference between largest and smallest word, by default the largest word is written with 10 times as large font than the smallest
scale	a fraction of the available space on figure that will be covered with the bounding boxes of words
min.freq	minimal frequency of words to be displayed
max.words	maximal number of words to be displayed
random.order	plot words in random order. If false, they will be plotted in decreasing frequency
colors	vector of colors fro the words. This vector will be extrapolated into as many colors as needed, starting with the first color for lower frequencies and ending with last color for higher frequencies.
random.colors	if true, assigns random color for the words.
fontface	fontface for the words (1 - regular; 2 - Bold; 3 - italic)
algorithm	algorithm to find positions of words possible values: "circle", "leftside" and "rightside".
tryfit	if TRUE the algorithm checks if all words fit to the figure, if not it tries gradually smaller values of scale parameter until everything fits
add	if TRUE adds the picture to existing plot.
grob	if TRUE returns the text grob instead of drawing it
dimensions	a two element vector of units giving the width and height of the word cloud respectively

Details

Uses the algorithm from wordcloud package to calculate the positions of the words. It then uses grid graphics to plot the words on screen. The shape of the wordcloud depends on the shape of the plotting window

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

Examples

```
plotWordcloud(c("Audi", "Volkswagen", "Opel", "Porsche", "Mercedez", "BMW"), 8:3)
```

`tissue_example` *Example gene expression dataset*

Description

`tissue_example` is a dataset extracted from Lukk *et al*, it contains a subset of 24 samples from more than 5000 in the original article. The gene expression in all the samples is measured using Affymetrix U133A array. The dataset is a list with 2 slots

1. `exp` - expression matrix;
2. `annot` - annotation data frame.

Details

The GOsummaries objects based on this data created in different examples are alaso attached to the package. These can be used to test the package if no internet connection is available. Names of these are `gs_kmeans`, `gs_limma`, `gs_limma_exp` and `gs_pca`

Author(s)

Raivo Kolde <raivo.kolde@eesti.ee>

References

Lukk M, Kapushesky M, Nikkila J, Parkinson H, Goncalves A, Huber W, Ukkonen E, Brazma A. "A global map of human gene expression." Nat Biotechnology. 2010 Apr;28(4):322-4.

Index