Package 'OmicInt'

October 28, 2021

Type Package

Title Omics Network Exploration

Version 1.1.7

Author Auste Kanapeckaite

Maintainer Auste Kanapeckaite <auste.kan@algorithm379.com>

Description Omics integration and detailed gene network exploration to identify expression patterns, prepare for pathway building, and find disease candidate genes; the package compliments research ``Insights into therapeutic targets and biomarkers using integrated multi-'omics' approaches for dilated and ischemic cardiomyopathies''; Auste Kanapeckaite and Neringa Burokiene; 2021, <doi:10.1093/intbio/zyab007>.

License GPL (>= 3)

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.1.2

Suggests viridis, utils, stats, graphics

Imports cluster, stringr,RCurl, ggplot2, mclust,gtools,methods ,tidyr,
dplyr,
tidyselect,pheatmap,reshape2,plotly,knitr,rmarkdown,lattice,
RColorBrewer, igraph, ggExtra, dendextend,STRINGdb

NeedsCompilation no

Repository CRAN

Date/Publication 2021-10-28 10:10:02 UTC

R topics documented:

`class_map` *class_map*

Description

Function provides visualisation of how the highest and lowest LFCscore genes cluster based on protein class data. Function requires a data frame generated by the score_genes function.

Usage

```
class_map(data, num = 0)
```

Arguments

`data`	Requires a data frame generated by score_genes; class - data frame
`num`	a number for genes to cluster from top upregulated and downregulated genes, if not selected all genes will be used; default 0, i.e. do not select a smaller set; class - integer

Value

dendogram; class - plot

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of class_map
df<-utils::read.table(path_to_test_data)
class_map(df)
## End(Not run)
```

class_summary *class_summary*

Description

class_summary function provides information on main protein classes. Barplot also helps to visualise the class distribution. Function depends on the data frame generated by score_genes function,

Usage

```
class_summary(data)
```

Arguments

`data`	Requires a data frame generated by score_genes; class - data frame

Value

multiple plots; class - plots

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of class_summary
df<-utils::read.table(path_to_test_data)
class_summary(df)
## End(Not run)
```

cluster_genes *cluster_genes*

Description

Function helps to select an optimal number of clusters and a model to be fitted during the EM phase of clustering for Gaussian Mixture Models. The function provides summaries and helps to visualise gene clusters based on generated data using score_genes function. Weighed gene expression is clustered based on the interactome complexity, i.e., the number of known interactors according to STRING DB, with a cutoff of 700 for the score threshold. The function also provides scatter plotting and dimension reduction plots to analyse the clusters and features in the experimental data.

Usage

```
cluster_genes(data, max_range = 20, clusters = NULL, modelNames = NULL)
```

Arguments

`data`	data frame containing processed expression file from score_genes with LFC-score; class - data frame
`max_range`	number of clusters to consider during model selection; default 20 clusters; class - integer
`clusters`	number of clusters to test not based on the best BIC output, user also needs to supply modelNames; class - integer
`modelNames`	can only be supplied when clusters are also specified, this option will model based on the user parameters; class - string

Value

A data frame object that contains a summary of clusters as well as clustering and summary plots

Examples

```
 ## Not run:
path_to_test_data<- system.file("extdata", "test_scores.tabular", package="OmicInt")
# basic usage of cluster_genes
df<-utils::read.table(path_to_test_data)
df<-cluster_genes(df)
head(df)

## End(Not run)
```

`cluster_heatmap` *cluster_heatmap*

Description

cluster_heatmap uses information mined from STRING database to map experimental, referenced, and inferred interactions to see if there are any interactors in the set of significantly changed genes. This heatmap provides clustered visualisation of all genes and the genes that have shared interactions.

Usage

```
cluster_heatmap(data)
```

Arguments

`data`	requires a data frame containing gene names as row names and a column with LFC values. Class - data frame

Value

heatmap; class - plot

Examples

```
 ## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of cluster_heatmap
df<-utils::read.table(path_to_test_data)
cluster_heatmap(df)

## End(Not run)
```

---

cluster_links *cluster_links*

Description

Function to select an optimal number of clusters and a model to be fitted during the EM phase of clustering for Gaussian Mixture Models. The function provides summaries and helps to visualise gene clusters based on generated data using score_genes function. Weighed gene expression is clustered based on a specific disease score which can be either the association or specificity for a disease, i.e., if the gene has known links to disease phenotypes or how specific it is when describing a pathology. The function also provides scatter plots and dimension reduction plots to analyse the clusters and features in the experimental data.

Usage

```
cluster_links(
  data,
  max_range = 20,
  type = "association",
  clusters = NULL,
  modelNames = NULL
)
```

Arguments

`data`	data frame containing processed expression file from score_genes with LFC-score; Class - data frame
`max_range`	number of clusters to consider during model selection; default 20 clusters. Class - integer
`type`	type of score to consider which can be either "association" or "specificity"; default "association". Class - string
`clusters`	number of clusters to test not based on the best BIC output, user also needs to supply modelNames; class - integer
`modelNames`	can only be supplied when clusters are also specified, this option will model based on the user parameters; class- string

Value

A data frame object that contains a summary of clusters; class - data frame

Examples

```
 ## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of cluster_links
df<-utils::read.table(path_to_test_data)
df<-cluster_links(df)
head(df)

## End(Not run)
```

CpG_summary *CpG_summary*

Description

CpG_summary function provides information on genes with CpG islands and GC content. The function checks genes against known CpG islands and provides various plots to assess emerging data features. The user can also specify if the plotting is necessary for location ("location") or protein class ("class"). Only genes with GC data are assessed.

Usage

```
CpG_summary(data, type = "class")
```

Arguments

`data`	Requires a data frame generated by score_genes; class - data frame
`type`	Requires to specify if plotting is performed for location or class types; default is "class". Alternatively, select "location". Class - string

Value

multiple plots (class - plots) and a data frame with GC content (class - data frame)

Examples

```
 ## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of CpG_summary
df<-utils::read.table(path_to_test_data)
return_df<-CpG_summary(df)
head(return_df)

## End(Not run)
```

`density_plot` *density_plot*

Description

Function plots a density plot for gene expression data prepared by the score_genes function. The plots can be used for a quick assessment of the overall gene expression distribution.

Usage

```
density_plot(data)
```

Arguments

`data`	Requires a data frame generated by score_genes; class - data frame

Value

multiple plots; class - plots

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of density_plot
df<-utils::read.table(path_to_test_data)
density_plot(df)
## End(Not run)
```

`feature_distribution` *feature_distribution*

Description

Function collects data from STRING database, scales, as well as prepares additional score integration to visualise data feature distribution.

Usage

```
feature_distribution(data)
```

Arguments

`data`	Requires a data frame generated by score_genes function; class - data frame

Value

multiple summary plots; class - plots

Examples

```
 ## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of feature_distribution
df<-utils::read.table(path_to_test_data)
feature_distribution(df)
## End(Not run)
```

HK_genes *HK_genes*

Description

HK_genes function provides a way to visualise how housekeeping genes changed throughout the conditions under the investigation. Depending on the number of conditions separate plots will be generated. Function requires a path variable to a normalised count data file.

Usage

```
HK_genes(data, meta)
```

Arguments

`data`	Requires a path variable to a data file of normalised scores in CSV format (comma separated); class - string
`meta`	Requires a path variable to a data file of metadata in CSV format (comma separated); class - string

Value

multiple plots; class - plots

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "normalised_counts.csv", package="OmicInt")
path_to_meta_data<- system.file("extdata", "meta_data.csv", package="OmicInt")
# basic usage of HK_genes
HK_genes(path_to_test_data,path_to_meta_data)
## End(Not run)
```

`interactor_map` *interactor_map*

Description

interactor_map uses information mined from STRING database to map experimental, predicted, or referenced interactions to see if there are any interactors in the set of significantly changed genes and how they are linked. The function requires a data frame prepared by score_genes. The output is a plot depicting interaction map.

Usage

```
interactor_map(data)
```

Arguments

`data`	requires a data frame containing gene names as row names and a column with LFC values; class - data frame

Value

interaction map/plot; class - plot

Examples

```
 ## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of interactor_map
df<-utils::read.table(path_to_test_data)
interactor_map(df)

## End(Not run)
```

`location_map` *location_map*

Description

Function provides visualisation of how the highest and lowest LFCscore genes cluster based on protein cellular location data. Function requires a data frame generated by the score_genes function.

Usage

```
location_map(data, num = 0)
```

Arguments

`data`	Requires a data frame generated by score_genes; class - data frame
`num`	a number for genes to cluster from top upregulated and downregulated genes, if not selected all genes will be used; default 0, i.e. do not select a smaller set; class - integer

Value

dendogram, class - plot

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of location_map
df<-utils::read.table(path_to_test_data)
location_map(df)
## End(Not run)
```

`location_summary` *location_summary*

Description

location_summary function provides information on main cellular locations. Barplot also helps to visualise the location data distribution.

Usage

```
location_summary(data)
```

Arguments

`data`	Requires a data frame generated by score_genes; class - data frame

Value

barplot; class - plot

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of location_summary
df<-utils::read.table(path_to_test_data)
location_summary(df)
## End(Not run)
```

`miRNA_network` *miRNA_network*

Description

miRNA_network function allows to assess how many genes are regulated by the same miRNA. Note if you supply too many genes the function will take longer to run.

Usage

```
miRNA_network(genes)
```

Arguments

`genes`	Requires a gene list (HGNC gene symbol); class list of strings

Value

a heatmap plot for found interactions and a list of miRNA and regulated genes. The list output value can be used for downstream analyses. Classes returned - a plot and a list

Examples

```
## Not run:
# basic usage of miRNA_network
return_df<-miRNA_network(c("PIP4K2A","MOB1A","PHACTR2","MDM2","YWHAG" ,"RAB31"  ))
head(return_df)

## End(Not run)
```

`miRNA_summary_predicted` *miRNA_summary_predicted*

Description

miRNA_summary_predicted function provides information on genes that have predicted/inferred miRNA regulating them. The function checks genes against predicted miRNA target database image and provides various plots to assess emerging data features. The user can also specify if the plotting is necessary for location ("location") or protein class ("class").

Usage

```
miRNA_summary_predicted(data, type = "class")
```

Arguments

`data`	Requires a data frame generated by score_genes. Class - data frame
`type`	Requires to specify a value for plotting. If plotting is performed for location select "location", alternatively select "class"; default is "class". Class - string

Value

a data frame with GC content; multiple plots summarising the data are also provided

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of miRNA_summary_predicted
df<-utils::read.table(path_to_test_data)
return_df<-miRNA_summary_predicted(df)
head(return_df)

## End(Not run)
```

miRNA_summary_validated *miRNA_summary_validated*

Description

miRNA_summary_validated function provides information on genes that have known miRNA regulating them. The function checks genes against known miRNA target database image and provides various plots to assess emerging data features. The user can also specify if the plotting is necessary for location ("location") or protein class ("class").

Usage

```
miRNA_summary_validated(data, type = "class")
```

Arguments

`data`	Requires a data frame generated by score_genes. Class - data frame
`type`	Requires to specify a value for plotting. If plotting is performed for location select "location", alternatively select "class"; default is "class". Class - string

Value

a data frame with GC content; multiple plots are also plotted summarising the data

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of miRNA_summary_validated
df<-utils::read.table(path_to_test_data)
return_df<-miRNA_summary_validated(df)
head(return_df)

## End(Not run)
```

`pattern_plots` *pattern_plots*

Description

pattern_plots function uses a subsetted pattern data from the function pattern_search. The function plots distribution plots as well as a selected set of genes and how they changed patterns. NOTE: if there are too many genes (>300), then individual gene expression plots will not be full because there is not enough colours in a palette to depict them all; in such a case, the function's violin plot can be used to assess the overall distribution and then the function should be repeated for filtered gene sets of interest to inspect individual expression values for each gene.

Usage

```
pattern_plots(data, meta, low = NA, high = NA, Condition = "Condition_1")
```

Arguments

`data`	Requires a data frame of normalised scores subsetted from pattern_search function. Class - string
`meta`	Requires a path variable to a data frame of metadata in CSV format. Class - string
`low`	the lowest value for the expression value; class - integer or float
`high`	the highest value for the expression value; class -integer or float
`Condition`	Requires a condition name to select if there are multiple conditions in meta data file, default "Condition_1". Conditions need to match between pattern_search and pattern_plot functions.Class - string

Value

function plots multiple plots, class - plots

Examples

```
## Not run:
path_to_test_data<- system.file("extdata","subsetted_data.csv", package="OmicInt")
path_to_meta_data<- system.file("extdata", "meta_data.csv", package="OmicInt")
# basic usage of pattern_search
pattern_plots(path_to_test_data,path_to_meta_data, 20, 10000)
## End(Not run)
```

`pattern_search` *pattern_search*

Description

pattern_search function searches for gene patterns that were upregulated or downregulated throughout the conditions when comparing to the geometric mean across all conditions. The geometric mean serves as a base value to compare across multiple conditions if more complex patterns exist and also allows for a universal baseline. Function takes path variables to data frames for normalised gene counts and meta data file (CSV format) as well as an additional variable that describes the name of a column that contains the condition under the investigation.

Usage

```
pattern_search(data, meta, Condition = "Condition_1")
```

Arguments

`data`	Requires a path variable to a data frame of normalised scores in CSV format; class - string
`meta`	Requires a path variable to a data frame of metadata in CSV format; class - string
`Condition`	Requires a condition name to select if there are multiple conditions in meta data file, default "Condition_1"; class - string

Value

a list variable which contains a pattern list with pattern names and associated genes; class - list

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "normalised_counts.csv", package="OmicInt")
path_to_meta_data<- system.file("extdata", "meta_data.csv", package="OmicInt")
# basic usage of pattern_search
pattern_search(path_to_test_data,path_to_meta_data)
## End(Not run)
```

`plot_3D_distribution` *plot_3D_distribution*

Description

Function allows to explore 3D distribution between the number of interactors, LFCscore and p.adj values. Function takes a data frame provided by score_genes function.

Usage

```
plot_3D_distribution(data, type = "association")
```

Arguments

`data`	a data frame containing processed expression file from score_genes with LFC-score; class - data frame
`type`	default value is "association", the user can select how to color data points depending on association or specificity score (e.g., selecting "specificity"); class - string

Value

function returns an interactive plot; class - plot

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "test_data.tabular", package="OmicInt")
# basic usage of plot_3D_distribution
df<-utils::read.table(path_to_test_data)
plot_3D_distribution(df)
## End(Not run)
```

`score_genes` *score_genes*

Description

Function collects data from STRINGDB and disease association databases to scale as well as prepare additional score integration. Function returns a data frame with calculated scores for downstream analyses.

Usage

```
score_genes(data, alpha = "association", beta = FALSE, gamma = FALSE)
```

Arguments

`data`	requires a path variable to CSV file containing gene names as row names and a column with LFC values in csv format (comma separated). Columns must contain values: 'Symbol', "log2FoldChange", and 'pvalue'. Class - data frame
`alpha`	default value returns "association" which gives a score from 0 to 1 based on how strongly the gene is associated with a disease or pathological phenotype; other options are "specificity" - to give values based on how specific the gene is for a given disease and "geometric" - to give a geometric score of both association and specificity. Class - string
`beta`	default FALSE; if TRUE, please supply data with column beta that contains information on gene associations from single cell studies. Class - string
`gamma`	default FALSE; if TRUE, please supply data with column gamma that contains information on gene associations from proteome studies. Class - string

Value

a data frame with calculated score values for the downstream analyses; class - data frame

Examples

```
## Not run:
path_to_test_data<- system.file("extdata", "data.csv", package="OmicInt")
#basic usage of score_genes function
df<-score_genes(path_to_test_data)
head(df)
## End(Not run)
```

Index