Package 'SingleCellSignalR'

April 27, 2024

Title Cell Signalling Using Single Cell RNAseq Data Analysis

Version 1.15.0

Author Simon Cabello-Aguilar Developer [aut],
Jacques Colinge Developer [aut, cre]

Maintainer Jacques Colinge Developer <jacques.colinge@umontpellier.fr>

Description
Allows single cell RNA seq data analysis, clustering, creates internal network and infers cell-cell interactions.

Depends R (>= 4.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports BiocManager, circlize, limma, igraph, gplots, grDevices, edgeR, data.table, pheatmap, stats, Rtsne, graphics, stringr, foreach, multtest, scran, utils,

RoxygenNote 7.2.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

biocViews SingleCell, Network, Clustering, RNASeq, Classification

git_url https://git.bioconductor.org/packages/SingleCellSignalR

git_branch devel

git_last_commit 299b500

git_last_commit_date 2023-10-24

Repository Bioconductor 3.19

Date/Publication 2024-04-26

Contents

`cell_classifier` *Cell classifier*

Description

Classifies cells using cell type specific markers.

Usage

```
cell_classifier(
  data,
  genes,
  markers = markers_default,
  tsne = NULL,
  plot.details = FALSE,
  write = TRUE,
  verbose = TRUE
)
```

Arguments

data	a data frame of n rows (genes) and m columns (cells) of read or UMI counts (note : rownames(data)=genes)
genes	a character vector of HUGO official gene symbols of length n
markers	a data frame of cell type signature genes
tsne	(optional) a table of n rows and 2 columns with t-SNE projection coordinates for each cell
plot.details	a logical (if TRUE, then plots the number of cells attributed to one cell type, see below)
write	a logical
verbose	a logical

Details

The ‘ markers‘ argument must be a table with cell type gene signatures, one cell type in each column. The column names are the names of the cell types.

The *markers.default* table provides an example of this format.

If ‘ tsne‘ is not provided, then the function will just not display the cells on the t-SNE. Although t-SNE maps are widely used to display cells on a 2D projection, the user can provide any table with two columns and a number of rows equal to the number of columns of ‘data‘ (e.g. the two first components of a PCA).

If ‘ plot.details‘ is TRUE, then the function plots the number of cells attributed to a single cell type as a function of the threshold applied to the normalized gene signature average.

If ‘ write‘ is TRUE, then the function writes four different text files. (1) The "raw classification matrix" provides the normalized average gene signature for each cell type in each individual cell, a number between 0 and 1. This matrix has one row per cell type and one column per cell, and the sum per column is 1. Row names are the cell type names (column names of the markers table) and the column names are the individual cell identifiers (column names of ‘data‘). (2) The "thresholded classification matrix", which is obtained by eliminating all the values of the "raw classification matrix" that are below a threshold a*. In practice, a* is automatically determined by the function to maximize the number of cells that are assigned to a single cell type and all the cells (columns) assigned to 0 or >1 cell types are discarded. The number of cells assigned to a single type depending on a* can be plotted by using the parameter ‘plot.details=TRUE‘. (3) A cluster vector assigning each cell to a cell type. Note that a supplementary, virtual cluster is created to collect all the cells assigned to 0 or >1 types. This virtual cluster is named "undefined". (4) A table associating each cell type to a cluster number in the cluster vector.

Value

The function returns a list containing the thresholded table, the maximum table, the raw table, a cluster vector and the cluster names. The maximum table is a special thresholded table where in every column only the maximum gene signature is kept. It can be used to force the classification of every cell.

Examples

```
data <- matrix(runif(1000,0,1),nrow=50,ncol=20)
rownames(data) <- paste("gene",seq_len(50))
markers <- matrix(paste("gene",seq_len(10)),ncol=5,nrow=2)
colnames(markers) <- paste("type",seq_len(5))
cell_classifier(data,rownames(data),markers)
```

`cell_signaling` *Cell Signaling*

Description

Computes "autocrine" or "paracrine" interactions between cell clusters.

Usage

```
cell_signaling(
  data,
  genes,
  cluster,
  int.type = c("paracrine", "autocrine"),
  c.names = NULL,
  s.score = 0.5,
  logFC = log2(1.5),
  species = c("homo sapiens", "mus musculus"),
  tol = 0,
  write = TRUE,
  verbose = TRUE
)
```

Arguments

`data`	a data frame of n rows (genes) and m columns (cells) of read or UMI counts (note : rownames(data)=genes)
`genes`	a character vector of HUGO official gene symbols of length n
`cluster`	a numeric vector of length m
`int.type`	"autocrine" or "paracrine"
`c.names`	(optional) cluster names
`s.score`	LRscore threshold
`logFC`	a number, the log fold-change threshold for differentially expressed genes
`species`	"homo sapiens" or "mus musculus"
`tol`	a tolerance parameter for balancing "autocrine|paracrine" interactions to the "autocrine" or "paracrine" group
`write`	a logical
`verbose`	a logical

Details

‘int.type‘ must be equal to "paracrine" or "autocrine" exclusively. The "paracrine" option looks for ligands expressed in cluster A and their associated receptors according to LR*db* that are expressed in any other cluster but A. These interactions are labelled "paracrine". The interactions that involve a ligand and a receptor, both differentially expressed in their respective cell clusters according to the **edgeR** analysis performed by the **cluster_analysis()** function, are labelled "specific". The "autocrine" option searches for ligands expressed in cell cluster A and their associated receptors also expressed in A. These interactions are labelled "autocrine". Additionally, it searches for those associated receptors in the other cell clusters (not A) to cover the part of the signaling that is "autocrine" and "paracrine" simultaneously. These interactions are labelled "autocrine/paracrine".

The ‘tol‘ argument allows the user to tolerate a fraction of the cells in cluster A to express the receptors in case ‘int.type="paracrine"‘, that is to call interactions that are dominantly paracrine though not exclusively. Conversely, it allows the user to reject interactions involving receptors that would be expressed by a small fraction of cluster A cells in case ‘int.type="autocrine"‘. By construction theassociation of these two options covers all the possible interactions and increasing the ‘tol‘ argument allows the user to move interactions from "autocrine" to "paracrine".

If the user does not set ‘c.names‘, the clusters will be named from 1 to the maximum number of clusters (cluster 1, cluster 2, ...). The user can exploit the ‘c.names‘ vector in the list returned by the **cell_classifier()** function for this purpose. The user can also provide her own cluster names.

‘s.score‘ is the threshold on the LRscore. The value must lie in the [0;1] interval, default is 0.5 to ensure confident ligand-receptor pair identifications (see our publication). Lower values increase the number of putative interactions while increasing the false positives. Higher values do the opposite.

‘logFC‘ is a threshold applied to the log fold-change (logFC) computed for each gene during the differential gene expression analysis. Its default value is log~2~(1.5) It further selects the differentially expressed genes (>logFC) after the p-value threshold imposed in the function **cluster_analysis()** below.

‘species‘ must be equal to "homo sapiens" or "mus musculus", default is "homo sapiens". In the case of mouse data, the function converts mouse genes in human orthologs (according to Ensembl) such that LR*db* can be exploited, and finally output genes are converted back to mouse.

If ‘write‘ is TRUE, then the function writes a text file that reports the interactions in the *cell-signaling* folder. This file is a 4-column table: ligands, receptors, interaction types ("paracrine", "autocrine", "autocrine/paracrine" and "specific"), and the associated LRscore.

Remarks:

- This function can be used with any ‘data‘ table associated with corresponding ‘genes‘ and ‘cluster‘ vectors, meaning that advanced users can perform their own data normalization and cell clustering upfront.

- In case the function **cluster_analysis()** was not executed, this function would work but "specific" interactions would not be annotated as such.

Value

The function returns "paracrine" or "autocrine" interaction lists. The interactions that are both "paracrine" and "autocrine" are annotated as "autocrine|paracrine" and are placed in the "autocrine" group by default.

Examples

```
data=matrix(runif(1000,0,1),nrow=5,ncol=200)
rownames(data) <- c("A2M","LRP1","AANAT","MTNR1A","ACE")
cluster=c(rep(1,100),rep(2,100))
cell_signaling(data,rownames(data),cluster,int.type="paracrine",write=FALSE)
```

`clustering` *Clustering*

Description

Identifies the cell clusters, i.e. the cell subpopulations.

Usage

```
clustering(
  data,
  n.cluster = 0,
  n = 10,
  method = c("kmeans", "simlr"),
  plot = TRUE,
  pdf = TRUE,
  write = TRUE
)
```

Arguments

`data`	a data frame of n rows (genes) and m columns (cells) of read or UMI counts (note : rownames(data)=genes)
`n.cluster`	a number, an estimation of the ideal number of clusters is computed if equal to 0
`n`	a number, the maximum to consider for an automatic determination of the ideal number of clusters
`method`	"kmeans" or "simlr"
`plot`	a logical
`pdf`	a logical
`write`	a logical

Details

If the user knows the number of clusters present in her data set, then ‘n.cluster‘ can be set and the estimation of the number of clusters is skipped. ‘n‘ is the maximum number of clusters that the automatic estimation of the number of clusters will consider. It is ignored if ‘n.cluster‘ is provided. ‘method‘ must be "simlr" or "kmeans" exclusively. If set to "simlr", then the function uses the **SIMLR()** function (**SIMLR** package) to perform clustering. If set to "kmeans" the

function will perform a dimensionality reduction by principal component analysis (PCA) followed by K-means clustering and 2-dimensional projection by t-distributed stochastic neighbor embedding (t-SNE). Regardless of the value of ‘method‘ ("simlr" or "kmeans"), in case ‘n.cluster‘ is not provided, then the function relies on the **SIMLR_Estimate_Number_of_Clusters()** function to determine the number of clusters, between 2 and ‘n‘. If ‘plot‘ is TRUE, then the function displays the t-SNE map with each cell colored according to the cluster it belongs to. If ‘method‘ argument is "simlr", then it further displays a heatmap of the similarity matrix calculated by the **SIMLR()** function. If ‘pdf‘ is TRUE, then the function exports the t-SNE plot in a pdf file in the *images* folder. The file is named "t-SNE_map-X.pdf", where X is the ‘method‘ argument. If ‘write‘ is TRUE, then the function writes two text files in the *data* folder. The first one is called "cluster-Y-X.txt", containing the cluster vector assigning each cell of ‘data‘ to a cluster. The second one is called "tsne-Y-X.txt", containing the coordinates of each cell in the 2D t-SNE projection. "X" is the ‘method‘ argument anf "Y" is the retained number of clusters.

Note that SIMLR might no longer be available in the most recent versions of R. It is thus necessary to load the library by yourself before calling this function if you want to use it (with `library(SIMLR)`).

Value

The function returns a list containing a numeric vector specifying the cluster assignment for each cell, a 2D t-SNE projection, and the number of cells per cluster.

Examples

```
data=matrix(runif(100000,0,1),nrow=500,ncol=200)
clustering(data,n.cluster=2,method="kmeans")
```

`cluster_analysis` *Cluster Analysis*

Description

Analysis of the differentially expressed genes in the clusters and their composition by a marker based approach.

Usage

```
cluster_analysis(
  data,
  genes,
  cluster,
  c.names = NULL,
  dif.exp = TRUE,
  s.pval = 10^-2,
  markers = NULL,
  write = TRUE,
  verbose = TRUE
)
```

Arguments

data	a data frame of n rows (genes) and m columns (cells) of read or UMI counts (note : rownames(data)=genes)
genes	a character vector of HUGO official gene symbols of length n
cluster	a numeric vector of length m
c.names	a vector of cluster names
dif.exp	a logical (if TRUE, then computes the diferential gene expression between the clusters using **edgeR**)
s.pval	a value, a fixed p-value threshold
markers	a table of cell type signature genes
write	a logical
verbose	a logical

Details

If ‘dif.exp‘ is TRUE, then the function uses **edgeR** functions **glmFit()** and **glmRT()** to find differentially expressed genes between one cluster and all the other columns of ‘data‘.

If ‘dif.exp‘ is FALSE, then the function skips the differential gene analysis.

If the user does not set ‘c.names‘, the clusters will be named from 1 to the maximum number of clusters (cluster 1, cluster 2, ...). The user can exploit the ‘c.names‘ vector in the list returned by the **cell_classifier()** function for this purpose. The user can also provide her own cluster names.

‘s.pval‘ is the adjusted (Benjamini-Hochberg) p-value threshold imposed to gene differential expression.

If ‘markers‘ is set, it must be a table with gene signatures for one cell type in each column. The column names are the names of the cell types.

If ‘markers‘ is not provided, then the function skips the cluster cell type calling step.

If ‘write‘ and ‘dif.exp‘ are both TRUE, then the function writes a text file named "table_dge_X.txt", where X is the cluster name, that contains the list of differentially expressed genes.

If ‘write‘ is TRUE and ‘markers‘ is provided, then the function writes in a second text file a table containing probabilities of assignments of each cluster to a cell type for each cell cluster. This cell type calling is performed as for the individual cells without thresholding but based on the cluster average transcriptome.

Remark: this function can be used with any ‘data‘ table associated with corresponding ‘genes‘ and ‘cluster‘ vectors, meaning that advanced users can perform their own data normalization and cell clustering upfront.

Value

The function returns a list comprised of a table of differentially expressed genes, a table of cell types, and a table of cell cluster types.

Examples

```
data=matrix(runif(1000,0,1),nrow=5,ncol=200)
rownames(data) <- c("A2M","LRP1","AANAT","MTNR1A","ACE")
cluster=c(rep(1,100),rep(2,100))
cluster_analysis(data,rownames(data),cluster,dif.exp=FALSE)
```

`data_prepare` *Data Prepare*

Description

Prepares the data for further analysis

Usage

```
data_prepare(
  file,
  most.variables = 0,
  lower = 0,
  upper = 0,
  normalize = TRUE,
  write = FALSE,
  verbose = TRUE,
  plot = FALSE
)
```

Arguments

`file`	a string for the scRNAseq data file
`most.variables`	a number
`lower`	a number in [0,1], low quantile threshold
`upper`	a number in [0,1], high quantile threshold
`normalize`	a logical, if TRUE, then computes 99th percentile normalization
`write`	a logical
`verbose`	a logical
`plot`	a logical

Details

'file' is the path to the file containing the read or UMI count matrix the user wants to analyze.

'most.variables' can be set to N to select the Nth most variables genes. This option allows the user to use a reduced matrix (N x number of cells) to perform the clustering step faster.

'lower' and 'upper' are used to remove the genes whose average counts are outliers. The values of these arguments are fractions of the total number of genes and hence must be between 0 and 1. Namely, if 'lower = 0.05', then the function removes the 5 removes the 5

If ‘normalize‘ is FALSE, then the function skips the 99th percentile normalization and the log transformation.

If ‘write‘ is TRUE, then the function writes two text files. One for the normalized and gene thresholded read counts table and another one for the genes that passed the lower and upper threshold. Note that the length of the genes vector written in the *genes.txt* file is equal to the number of rows of the table of read counts written in the *data.txt* file.

Value

The function returns a data frame of filtered and/or normalized data with genes as row names.

Examples

```
file <- system.file("scRNAseq_dataset.txt",package = "SingleCellSignalR")
data <- data_prepare(file = file)
```

`example_dataset` *Example dataset*

Description

Example dataset

Usage

```
example_dataset
```

Format

A data frame with 1520 rows of 401 variables:

Source

EDFR\&D

`expression_plot` *Expression Plot*

Description

Displays the level of expression of a gene in each cell on the 2D projected data.

Usage

```
expression_plot(data, name, tsne, colors = c("default", "rainbow", "heat"))
```

Arguments

`data`	a data frame of n rows (genes) and m columns (cells) of read or UMI counts (note : rownames(data)=genes)
`name`	the identifier of the gene of interest
`tsne`	a table of n rows and 2 columns with 2D projection coordinates for each cell
`colors`	"default" returns the default colorpanel, also accepts "rainbow" or "heat"

Details

This function displays the expression level of a gene of interest on a 2D projection.

‘name‘ can be any character that corresponds to a row name of ‘data‘.

‘tsne‘ corresponds to the 2D coordinates for each cell. Although t-SNE maps are widely used to display cells on a 2D projection, the user can provide any table with two columns and a number of rows equal to the number of columns of *data* (i.e. the two first components of a PCA).

‘colors‘ must be "default", "rainbow" or "heat" exclusively. "rainbow" and "heat" are the color palettes provided in R.

Value

The function returns a R plot.

Examples

```
data <- matrix(runif(5,0,1),ncol=5)
data[2] <- data[5] <- 0
rownames(data) <- "gene 1"
tsne <- matrix(runif(10,0,1),ncol=2)
expression_plot(data,"gene 1",tsne)
```

`expression_plot_2` *Expression Plot 2*

Description

Displays the level of expression of two genes in each cell on the 2D projected data.

Usage

```
expression_plot_2(data, name.1, name.2, tsne)
```

Arguments

`data`	a data frame of n rows (genes) and m columns (cells) of read or UMI counts (note : rownames(data)=genes)
`name.1`	the identifier of the first gene of interest
`name.2`	the identifier of the second gene of interest
`tsne`	a table of n rows and 2 columns with t-SNE projection coordinates for each cell

Details

This function can be used independantly from any other. It displays the expression level of two genes of interest on a 2D projection.

‘name.1‘ and ‘name.2‘ can be any characters that correspond to a row name of ‘data‘.

Value

The function returns a R plot.

Examples

```
data <- matrix(runif(100,0,1),nrow=2,ncol=50)
rownames(data) <- c("gene 1", "gene 2")
tsne <- matrix(runif(100,-1,1),ncol=2)
expression_plot_2(data,"gene 1","gene 2",tsne)
```

inter_network *inter network*

Description

Computes intercellular gene networks.

Usage

```
inter_network(
  data,
  genes,
  cluster,
  signal,
  c.names = NULL,
  species = c("homo sapiens", "mus musculus"),
  write = TRUE,
  plot = FALSE,
  verbose = TRUE
)
```

Arguments

data	a data frame of n rows (genes) and m columns (cells) of read or UMI counts (note : rownames(data)=genes)
genes	a character vector of HUGO official gene symbols of length n
cluster	a numeric vector of length m
signal	a list (result of the **cell_signaling()** function)
c.names	(optional) cluster names
species	"homo sapiens" or "mus musculus"
write	a logical (if TRUE writes graphML and text files for the interface and internal networks)
plot	a logical
verbose	a logical

Details

'signal' is a list containing the cell-cell interaction tables. It is the result of the **cell_signaling()** function.

If the user does not set 'c.names', the clusters will be named from 1 to the maximum number of clusters (cluster 1, cluster 2, ...). The user can exploit the 'c.names' vector in the list returned by the **cell_classifier()** function for this purpose. The user can also provide her own cluster names.

‘species‘ must be equal to "homo sapiens" or "mus musculus". In the case of mouse data, the function converts mouse genes in human orthologs (according to Ensembl) such that the Reactome/KEGG interaction database can be exploited, and finally output genes are converted back to mouse.

If ‘write‘ is TRUE, then the function writes four different files. A graphML file in the *cell-signaling* folder for intercellular interactions between each pair of clusters named "intercell_network_Z~1~-Z~2~.graphml", where Z~1~ and Z~2~ are the *c.names* of the clusters. A graphML file in the *cell-signaling* folder that contains a compilation of all the intercellular, ligand-receptor interactions named "full-intercellular-network.graphml". A text and a graphML file in the *networks* folder containing the intracellular network for each cell cluster named "intracell_network_Z.txt" and "intracell_network_Z.graphml", where Z is the *c.names* of the cluster.

Value

The function returns a list containing the tables of interaction between two cell types and the table for the full network of all the cell types.

Examples

```
m <- data.frame(cell.1=runif(10,0,2),cell.2=runif(10,0,2),cell.3=runif(10,0,2),
cell.4 <- runif(10,0,2),cell.5=runif(10,0,2),cell.6=runif(10,0,2),cell.7=
runif(10,0,2))
rownames(m) <- paste("gene", seq_len(10))
cluster <- c(1,1,1,2,3,3,2)
inter_network(m,rownames(m),cluster,signal=NULL)
```

---

intra_network *intra network*

Description

Computes intracellular networks linked to genes of interest.

Usage

```
intra_network(
  goi,
  data,
  genes,
  cluster,
  coi,
  cell.prop = 0.2,
  c.names = NULL,
  signal = NULL,
  write = TRUE,
  plot = TRUE,
  add.lig = TRUE,
  species = c("homo sapiens", "mus musculus"),
```

```
  connected = FALSE,
  verbose = TRUE
)
```

Arguments

goi	gene of interest (typically a receptor)
data	a data frame of n rows (genes) and m columns (cells) of read or UMI counts (note : rownames(data)=genes)
genes	a character vector of HUGO official gene symbols of length n
cluster	a numeric vector of length m
coi	name of the cluster of interest
cell.prop	a threshold, only the genes expressed in this proportion of the cells of the coi will be taken into account
c.names	(optional) cluster names
signal	(optional) a list (result of the **cell_signaling()** function)
write	a logical (if TRUE writes graphML and text files for the internal networks)
plot	a logical
add.lig	a logical (if TRUE adds the goi associated ligands from signal to the network)
species	"homo sapiens" or "mus musculus"
connected	a logical (if TRUE keeps only the genes connected to the goi)
verbose	a logical

Details

'signal' is a list containing the cell-cell interaction tables. It is the result of the **cell_signaling()** function.

'cell.prop' is set to 0.2 by default to avoid unreadable downstream networks. However if the calculated network is too small or non-existent (or too big) the user can try lower (or higher) values.

If the user does not set 'c.names', the clusters will be named from 1 to the maximum number of clusters (cluster 1, cluster 2, ...). The user can exploit the 'c.names' vector in the list returned by the **cell_classifier()** function for this purpose. The user can also provide her own cluster names.

'species' must be equal to "homo sapiens" or "mus musculus". In the case of mouse data, the function converts mouse genes in human orthologs (according to Ensembl) such that the Reactome/KEGG interaction database can be exploited, and finally output genes are converted back to mouse.

If 'write' is TRUE, then the function writes two different files. A graphML file in the *network* folder for intracellular interactions downstream the gene of interest (goi) named "intracell_network_coi-receptors.graphml". A text file in the *network* folder containing the information about the pathways in which the interactions are in, named "intracell_network_pathway_analysis_coi-receptors.txt".

Value

The function returns a list containing the internal networks linked to the genes of interest (goi)

Examples

```
data <- matrix(runif(1000,0,1),nrow=5,ncol=200)
genes <- c("A2M","LRP1","AANAT","MTNR1A","ACE")
cluster <- c(rep(1,100),rep(2,100))
intra_network(goi=c("TGFBR1","ERBB2"),data,genes,cluster,coi="cluster 1")
```

LRdb *Ligand/Receptor interactions data table*

Description

Ligand/Receptor interactions data table

Usage

```
LRdb
```

Format

A data frame with 3251 rows of 13 variables:

ligand ligand gene symbol

receptor receptor gene symbol

source provenance of the interaction

PMIDs PubmedID of the publication reporting the interaction ...

Source

EDFR\&D

LRscore *Calculation of the LRscore*

Description

Calculation of the LRscore

Usage

```
LRscore(l, r, s)
```

Arguments

l	a value (or a vector) of the mean ligand expression in the secreting cluster
r	a value (or a vector) of the mean receptor expression in the receiving cluster
s	a value for scaling the score (usually the mean of the whole read count table, the median or another similar value is possible), must be over 0

Value

a value or a vector

Examples

```
l=1
r=9
s=5
LRscore(l,r,s)
```

`markers` *Markers*

Description

Provides a table of cell type specific markers.

Usage

```
markers(category = c("immune", "tme", "melanoma", "bc"))
```

Arguments

`category`	one or several of the following "immune", "tme", "melanoma", "bc"

Details

To use this function the user must have some knowledge about the cell composition of her data set. For instance, if the dataset comes from a breast cancer tumor, the user may select "bc", but also "immune" and "tme" for cells of the microenvironment (see **Examples of use** in the User's Guide vignette).

Value

The function returns a cell type specific markers table.

Examples

```
markers(c("immune","bc"))
```

`markers_default` *A list of cell types markers*

Description

A list of cell types markers

Usage

```
markers_default
```

Format

A data frame with 95 rows of 15 variables:

TNBC Triple Negative Breast Cancer markers

HER+ HER+ Breast Cancer markers

ER+ ER+ Breast Cancer markers

T-cells T cells markers

B-cells B cells markers

Macrophages Macrophages markers

Endothelial cells Endothelial cells markers

CAFs Cancer Associated Fibroblasts markers

melanoma Melanoma markers

Cytotoxic cells Cytotoxic cells markers

DC Dendritic cells markers

Mast cells Mastocytes cells markers

Neutrophils Neutrophils cells markers

NK cells Natural Killer cells markers

Treg Regulatory T-cells markers ...

Source

EDFR\&D

mm2Hs *Mus Musculus (mm) to Homo Sapiense (Hs) Orthology table*

Description

Mus Musculus (mm) to Homo Sapiense (Hs) Orthology table

Usage

```
mm2Hs
```

Format

A data frame with 16519 rows of 2 variables:

Mouse gene name mm gene symbol

Gene name Hs gene symbol ...

Source

EDFR\&D

mv_interactions *most variable interactions*

Description

Displays a heatmap showing the most variable interactions over all clusters.

Usage

```
mv_interactions(
  data,
  genes,
  cluster,
  c.names = NULL,
  n = 30,
  species = c("homo sapiens", "mus musculus")
)
```

Arguments

data	a data frame of n rows (genes) and m columns (cells) of read or UMI counts (note : rownames(data)=genes)
genes	a character vector of HUGO official gene symbols of length n
cluster	a numeric vector of length m
c.names	(optional) cluster names
n	an integer the number of most variables interactions
species	"homo sapiens" or "mus musculus"

Value

The function displays a heatmap showing the most variable interactions over all clusters

Examples

```
data <- matrix(runif(1000,0,1),nrow=5,ncol=200)
genes <- c("gene 1","gene 2","gene 3","gene 4","gene 5")
cluster <- c(rep(1,100),rep(2,100))
mv_interactions(data,genes,cluster)
```

PwC_ReactomeKEGG *Pathway Commons Reactome KEGG 2019-05-08*

Description

Pathway Commons Reactome KEGG 2019-05-08

Usage

```
PwC_ReactomeKEGG
```

Format

A data frame with 26067 rows of 6 variables:

a.gn interactant 1 gene symbol

b.gn interactant 2 gene symbol

type interaction type

pathway Associated pathway ...

Source

EDFR\&D

simplify_interactions *simplify_interactions*

Description

simplify_interactions

Usage

```
simplify_interactions(t, lr = NULL, autocrine = FALSE)
```

Arguments

t	the network to be simplified
lr	ligand receptor interactions
autocrine	a logical

Value

t

Examples

```
t=data.frame(a.gn=c("CEP63","CEP63"),b.gn=c("MZT2A","DYNC1L2"),
type=c("in-complex-with","in-complex-with"))
simplify_interactions(t)
```

visualize_interactions
Visualize interactions

Description

Creates chord diagrams from the interactions tables.

Usage

```
visualize_interactions(
  signal,
  show.in = NULL,
  write.in = NULL,
  write.out = FALSE,
  method = "default",
  limit = 30
)
```

Arguments

signal	a list of data frames result of the **cell_signaling()** function
show.in	a vector of which elements of "'signal"' must be shown
write.in	a vector of which elements of "'signal"' must be written
write.out	a logical
method	a string (usually relative to the experiment)
limit	a value between 1 and number of interactions

Details

'show.in' gives the elements of 'signal' to be displayed in the plot window.

'write.in' gives the elements of 'signal' to be written as pdf files in the *images* folder.

If 'write.out' is TRUE, then the function writes a pdf file with a summary of the all the interactions of 'signal' as a chord diagram.

'limit' is the maximum number of interactions displayed on one chord diagram. Raising this limit over 30 may decrease the visibility.

Value

The function returns images in the plot window of Rstudio and images in the pdf format in the *images* folder.

Examples

```
int.1 <- matrix(c("gene 1","gene 1", "gene 2", "gene 3"),ncol=2)
colnames(int.1) <- c("cluster 1","cluster 2" )
int.2 <- matrix(c("gene 1","gene 4","gene 4","gene 2","gene 3","gene 3"),
ncol=2)
colnames(int.2) <- c("cluster 1","cluster 3" )
signal <- list(int.1,int.2)
names(signal) <- c("1-2","1-3")
visualize_interactions(signal)
```

Index