Package ‘InterCellar’

May 3, 2024

Title InterCellar: an R-Shiny app for interactive analysis and exploration of cell-cell communication in single-cell transcriptomics

Version 2.10.0

Description InterCellar is implemented as an R/Bioconductor Package containing a Shiny app that allows users to interactively analyze cell-cell communication from scRNA-seq data. Starting from precomputed ligand-receptor interactions, InterCellar provides filtering options, annotations and multiple visualizations to explore clusters, genes and functions. Finally, based on functional annotation from Gene Ontology and pathway databases, InterCellar implements data-driven analyses to investigate cell-cell communication in one or multiple conditions.

License MIT + file LICENSE

Imports config, golem, shiny, DT, shinydashboard, shinyFiles, shinycssloaders, data.table, fs, dplyr, tidyr, circlize, colourpicker, dendextend, factoextra, ggplot2, plotly, plyr, shinyFeedback, shinyalert, tibble, umap, visNetwork, wordcloud2, readxl, htmlwidgets, colorspace, signal, scales, htmltools, ComplexHeatmap, grDevices, stats, tools, utils, biomaRt, rlang, fmsb, igraph

Encoding UTF-8

RoxygenNote 7.1.1

Suggests testthat (>= 3.0.0), knitr, rmarkdown, glue, graphite, processx, attempt, BiocStyle, htttr

Config/testthat/edition 3

URL https://github.com/martaint/InterCellar

BugReports https://github.com/martaint/InterCellar/issues

VignetteBuilder knitr

biocViews Software, SingleCell, Visualization, GO, Transcriptomics

Depends R (>= 4.1)

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

git_branch RELEASE_3_19
## Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>annotateGO</td>
<td>3</td>
</tr>
<tr>
<td>annotatePathways</td>
<td>4</td>
</tr>
<tr>
<td>buildPairsbyFunctionMatrix</td>
<td>4</td>
</tr>
<tr>
<td>checkLL_RR</td>
<td>5</td>
</tr>
<tr>
<td>circlePlot</td>
<td>5</td>
</tr>
<tr>
<td>combineAnnotations</td>
<td>6</td>
</tr>
<tr>
<td>createBarPlot1_ggplot</td>
<td>6</td>
</tr>
<tr>
<td>createBarPlot2_CV</td>
<td>7</td>
</tr>
<tr>
<td>createBarPlot2_ggplot</td>
<td>7</td>
</tr>
<tr>
<td>createBarPlot_CV</td>
<td>8</td>
</tr>
<tr>
<td>createNetwork</td>
<td>9</td>
</tr>
<tr>
<td>dendroIntPairModules</td>
<td>9</td>
</tr>
<tr>
<td>elbowPoint</td>
<td>10</td>
</tr>
<tr>
<td>ensemblLink</td>
<td>10</td>
</tr>
<tr>
<td>getBack2BackBarplot</td>
<td>11</td>
</tr>
<tr>
<td>getBarplotDF</td>
<td>11</td>
</tr>
<tr>
<td>getBarplotDF2</td>
<td>12</td>
</tr>
<tr>
<td>getClusterA_Names</td>
<td>12</td>
</tr>
<tr>
<td>getClusterColors</td>
<td>13</td>
</tr>
<tr>
<td>getClusterNames</td>
<td>13</td>
</tr>
<tr>
<td>getClusterNetwork</td>
<td>14</td>
</tr>
<tr>
<td>getClusterSize</td>
<td>14</td>
</tr>
<tr>
<td>getDistinctCouplets</td>
<td>15</td>
</tr>
<tr>
<td>getDotPlotSelInt</td>
<td>15</td>
</tr>
<tr>
<td>getGeneTable</td>
<td>16</td>
</tr>
<tr>
<td>getGObiomaRt</td>
<td>17</td>
</tr>
<tr>
<td>getHitsf</td>
<td>17</td>
</tr>
<tr>
<td>getIntFlow</td>
<td>18</td>
</tr>
<tr>
<td>getNtermsBYdb</td>
<td>18</td>
</tr>
<tr>
<td>getNumLR</td>
<td>19</td>
</tr>
<tr>
<td>getPieChart</td>
<td>19</td>
</tr>
<tr>
<td>getRadar_df</td>
<td>20</td>
</tr>
<tr>
<td>getRankedTerms</td>
<td>22</td>
</tr>
<tr>
<td>getSignificantFunctions</td>
<td>22</td>
</tr>
<tr>
<td>getSignificantFunctions_multiCond</td>
<td>23</td>
</tr>
<tr>
<td>getSignif_table</td>
<td>23</td>
</tr>
<tr>
<td>getSunburst</td>
<td>24</td>
</tr>
</tbody>
</table>
annotateGO

Perform GO annotation of input data

Description

Perform GO annotation of input data

Usage

annotateGO(
  input_select_ensembl,
  input_go_evidence_exclude,
  input_go_sources_checkbox,
  input.data
)

Arguments

input_select_ensembl
  ensembl version selected by user
input_go_evidence_exclude
  evidence codes to exclude by user
input_go_sources_checkbox
  GO sources to use by user
input.data
  preprocessed input data

Value

GO_annotation
annotatePathways  
Annotate pathways for input data

Description
Annotate pathways for input data

Usage
annotatePathways(selected.db, input.data)

Arguments
- selected.db: pathways sources to use
- input.data: filtered input data

Value
pathways_annotation

buildPairsbyFunctionMatrix
Build binary matrix with int-pairs in rows, functions in cols

Description
Build binary matrix with int-pairs in rows, functions in cols

Usage
buildPairsbyFunctionMatrix(functions_df)

Arguments
- functions_df: annotated df (GO/path/combined)

Value
binary matrix
**checkLL_RR**

Manually change the annotation of L-L and R-R pairs

**Description**

Manually change the annotation of L-L and R-R pairs

**Usage**

```r
checkLL_RR(input.data)
```

**Arguments**

- `input.data`:
  - preprocessed table

**Value**

`input.data`

**Examples**

```r
data(input.data)
checked.input.data <- checkLL_RR(input.data)
```

---

**circlePlot**

Plot circle plot

**Description**

Plot circle plot

**Usage**

```r
circlePlot(data, cluster_colors, ipm_color, int_flow, link.color)
```

**Arguments**

- `data`:
  - subset of input data by flow / intpair module
- `cluster_colors`:
  - global
- `ipm_color`:
  - single color for chosen int-pair module
- `int_flow`:
  - string specifying the flow
- `link.color`:
  - string specifying variable by which to color links

**Value**

`circle plot`
combineAnnotations  

Combine GO annotation and pathways in a unique object

Description

Combine GO annotation and pathways in a unique object

Usage

combineAnnotations(GO_annotation, pathways_annotation)

Arguments

- GO_annotation: data
- pathways_annotation: data

Value

combined annotation dataframe

createBarPlot1_ggplot  

Create ggplot barplot to be saved in tiff

Description

Create ggplot barplot to be saved in tiff

Usage

createBarPlot1_ggplot(
  barplotDF,
  input_cluster_selected_checkbox,
  input_num_or_weight_bar1
)

Arguments

- barplotDF: dataframe with N interactions per cluster (auto/para)
- input_cluster_selected_checkbox: checkbox input
- input_num_or_weight_bar1: number of int or weighted number by score

Value

ggplot barplot
createBarPlot2.CV  

Create barplot of number of interaction for selected cluster

Description

Create barplot of number of interaction for selected cluster

Usage

createBarPlot2.CV(
  barplotDF2,
  input_cluster_selected_checkbox,
  input_clust_barplot2
)

Arguments

barplotDF2  dataframe with barplot data
input_cluster_selected_checkbox  selected clusters to keep
input_clust_barplot2  selected cluster to plot

Value

plotly fig

createBarPlot2.ggplot  

Create ggplot barplot of Nint per cluster selected

Description

Create ggplot barplot of Nint per cluster selected

Usage

createBarPlot2.ggplot(
  barplotDF2,
  input_cluster_selected_checkbox,
  input_clust_barplot2
)
createBarPlot_CV

Arguments

barplotDF2 dataframe with barplot data
input_cluster_selected_checkbox selected clusters to keep
input_clust_barplot2 selected cluster to plot

Value

ggplot barplot

createBarPlot_CV Create Barplot cluster-verse

Description

Create Barplot cluster-verse

Usage

createBarPlot_CV(
    barplotDF,
    input_cluster_selected_checkbox,
    input_num_or_weight_bar1
)

Arguments

barplotDF dataframe with N interactions per cluster (auto/para)
input_cluster_selected_checkbox checkbox input
input_num_or_weight_bar1 number of int or weighted number by score

Value

plotly barplot
**createNetwork**  
*Create Network of clusters*

**Description**  
Create Network of clusters

**Usage**  
createNetwork(data.filt.cluster, input_num_or_weight_radio, input_edge_weight)

**Arguments**  
data.filt.cluster  
filtered input data (by clusters)  
input_num_or_weight_radio  
either number of interactions or weighted by score  
input_edge_weight  
small, medium or large from user input

**Value**  
list containing nodes and edges for network

---

**dendroIntPairModules**  
*Get dendrogram of int pair modules*

**Description**  
Get dendrogram of int pair modules

**Usage**  
dendroIntPairModules(pairs_func_matrix)

**Arguments**  
pairs_func_matrix  
binary matrix pairs x functions

**Value**  
list with dendrogram, hclust and umap
elbowPoint

*Determin the elbow point on a curve (from package akmedoids)*

**Description**

Given a list of x, y coordinates on a curve, function determines the elbow point of the curve.

**Usage**

`elbowPoint(x, y)`

**Arguments**

- `x` vector of x coordinates of points on the curve
- `y` vector of y coordinates of points on the curve

**Details**

highlight the maximum curvature to identify the elbow point (credit: `github.com/agentlans`)

**Value**

an x, y coordinates of the elbow point.

---

ensemblLink

*Get html link to ensembl*

**Description**

Get html link to ensembl

**Usage**

`ensemblLink(ensembl)`

**Arguments**

- `ensembl` symbol

**Value**

html link to website
getBack2BackBarplot  
Get back-to-back barplot for 2 conditions comparison

Description
Get back-to-back barplot for 2 conditions comparison

Usage
getBack2BackBarplot(tab_c1, tab_c2, lab_c1, lab_c2)

Arguments
- tab_c1: barplot dataframe generated by getBarplotDF() for condition 1
- tab_c2: barplot dataframe generated by getBarplotDF() for condition 1
- lab_c1: label for condition 1
- lab_c2: label for condition 2

Value
ggplot object

getBarplotDF  
Get dataframe for plotting barplot (all clusters)

Description
Get dataframe for plotting barplot (all clusters)

Usage
getBarplotDF(
  data.filt.bar,
  input_cluster_selected_checkbox,
  input_num_or_weight_bar1
)

Arguments
- data.filt.bar: filtered object (checkbox auto/para)
- input_cluster_selected_checkbox: checkbox input
- input_num_or_weight_bar1: number of int or weighted number by score
getBarplotDF2  
*Get dataframe for barplot (by cluster)*

**Description**
Get dataframe for barplot (by cluster)

**Usage**
```r
getBarplotDF2(filt.data, input_cluster_selected_checkbox, input_clust_barplot2)
```

**Arguments**
- `filt.data`: input data filtered in cluster-verse
- `input_cluster_selected_checkbox`: selected clusters to keep
- `input_clust_barplot2`: selected cluster to plot

**Value**
- dataframe with num int per cluster

getClusterA_Names  
*Get cluster names only from sender cluster A*

**Description**
Get cluster names only from sender cluster A

**Usage**
```r
getClusterA_Names(input.data)
```

**Arguments**
- `input.data`: preprocessed input data

**Value**
- named list of clusters
getClusterColors

*Get colors for clusters*

**Description**

Get colors for clusters

**Usage**

`getClusterColors(input.data)`

**Arguments**

- `input.data` : preprocessed input data

**Value**

named vector with colors per cluster

---

getClusterNames

*Get clusters names from initial input data*

**Description**

Get clusters names from initial input data

**Usage**

`getClusterNames(input.data)`

**Arguments**

- `input.data` : preprocessed input data

**Value**

named list of clusters

**Examples**

```r
data(input.data)
cluster_list <- getClusterNames(input.data)
```
**getClusterNetwork**

*Creating edges dataframe for network of clusters*

**Description**

Creating edges dataframe for network of clusters

**Usage**

```r
getClusterNetwork(input.data, input_num_or_weight_radio, input_edge_weight)
```

**Arguments**

- `input.data`: preprocessed input data
- `input_num_or_weight_radio`: either num of interactions or weighted by score
- `input_edge_weight`: small, medium or large from user input

**Value**

edges dataframe

---

**getClusterSize**

*Get Clusters size*

**Description**

Get Clusters size

**Usage**

```r
getClusterSize(cl, edges.df, input_num_or_weight_radio)
```

**Arguments**

- `cl`: cluster name
- `edges.df`: dataframe with edges for network
- `input_num_or_weight_radio`: either num of interactions or weighted by score

**Value**

sum of n interactions or weighted num for that cluster
getDistinctCouplets

Description
Get table of unique int-pairs/clust-pairs couplets

Usage
getDistinctCouplets(
  data_cond1,
  data_cond2,
  data_cond3 = NULL,
  lab_c1,
  lab_c2,
  lab_c3 = NULL
)

Arguments
- data_cond1: filt.data() corresponding to chosen condition 1
- data_cond2: filt.data() corresponding to chosen condition 2
- data_cond3: filt.data() corresponding to chosen condition 3
- lab_c1: data label for condition 1
- lab_c2: data label for condition 2
- lab_c3: data label for condition 3

Value
modified filt.data containing only unique couplets

getDotPlot_selInt

Description
Functions to plot DotPlots

Usage
gDotPlot_selInt(
  selected_tab,
  clust.order,
  low_color = "aquamarine",
  high_color = "#131780"
)
getGeneTable

**Arguments**

- `selected_tab`  
  selected rows of filt.data by selection from gene table
- `clust.order`  
  how to order clusters
- `low_color`  
  of dotplot
- `high_color`  
  of dotplot

**Value**

list with modified selected data and ggplot2 dotplot

---

**Description**

Get table for gene-verse

**Usage**

```
getGeneTable(input.data)
```

**Arguments**

- `input.data`  
  preprocessed input data

**Value**

gene table with unique intpairs (no connection to clusters)

**Examples**

```
data(input.data)
gene_table <- getGeneTable(input.data)
```
getGObiomaRt

Connection to Ensembl via biomaRt to get GO terms

Description
Connection to Ensembl via biomaRt to get GO terms

Usage
getGObiomaRt(input_select_ensembl, input.data)

Arguments
- input_select_ensembl
  chosen version of Ensembl
- input.data
  filtered input data

Value
dataframe with GO annotation

getHitsf
Subfunction to calculate significant functions by permutation test

Description
Subfunction to calculate significant functions by permutation test

Usage
getHitsf(mat, gpModules_assign)

Arguments
- mat
  binary matrix of functional terms by int-pairs
- gpModules_assign
  assignment of intpairs to modules

Value
matrix with hits

Example
**getIntervalFlow**

*Get subset of interactions corresponding to a certain viewpoint and flow*

**Description**

Get subset of interactions corresponding to a certain viewpoint and flow.

**Usage**

`getIntFlow(vp, input.data, flow)`

**Arguments**

- `vp`: viewpoint cluster
- `input.data`: preprocessed/filtered input data
- `flow`: one among `directed_out`, `directed_in` or `undirected`

**Value**

subset of data

**Examples**

```r
data(input.data)
caf_out <- getIntFlow(vp = "CAF", input.data, flow = "directed_out")```

---

**getNtermsBYdb**

*Calculate number of terms of a database*

**Description**

Calculate number of terms of a database.

**Usage**

`getNtermsBYdb(annotation)`

**Arguments**

- `annotation`: data from either pathways, GO or combined

**Value**

number of terms by dataset
**getNumLR**

*Get number of unique ligands and receptors*

**Description**

Get number of unique ligands and receptors

**Usage**

```r
ggetNumLR(gene.table, type)
```

**Arguments**

- `gene.table`: gene table of unique int-pairs
- `type`: either L or R

**Value**

number of L or R genes

---

**getPieChart**

*Get Pie Chart of unique couplets*

**Description**

Get Pie Chart of unique couplets

**Usage**

```r
ggetPieChart(data_dotplot)
```

**Arguments**

- `data_dotplot`: same data used to generate dotplot

**Value**

pie chart
getRadar_df

# Get radar plot of relative numbers of interactions for a certain cell type
# @param tab_c1 barplot dataframe from Viewpoint generated by getBarplotDF2() containing data for condition 1
# @param tab_c2 barplot dataframe from Viewpoint generated by getBarplotDF2() containing data for condition 2
# @param tab_c3 barplot dataframe from Viewpoint generated by getBarplotDF2() containing data for condition 3
# @param lab_c1 label for condition 1
# @param lab_c2 label for condition 2
# @param lab_c3 label for condition 3
# @param cell_name label of cell type of interest
# @return plot
@importFrom fmsb radarchart
@importFrom data.table transpose
getRadarPlot <- function(tab_c1, tab_c2, tab_c3, lab_c1, lab_c2, lab_c3, cell_name)
if(is.null(tab_c3))
  df <- merge(tab_c1, tab_c2, by = "Clusters", all = TRUE)
  colnames(df) <- c("Clusters", "nint_c1", "nint_c2")
else
  df <- merge(tab_c1, tab_c2, by = "Clusters", all = TRUE)
  df <- merge(df, tab_c3, by = "Clusters", all = TRUE)
  colnames(df) <- c("Clusters", "nint_c1", "nint_c2", "nint_c3")

  if(is.null(lab_c3))
    rownames(radar_df) <- c("max", "min", lab_c1, lab_c2)
  else
    rownames(radar_df) <- c("max", "min", lab_c1, lab_c2, lab_c3)

colnames(radar_df) <- cluster_names

color <- c("#438ECC", "#E97778", "#00BA38")

fmsb::radarchart( radar_df, axistype = 1,
  polygon.pcol = color, pfcol = scales::alpha(color, 0.5),
  plwd = 2, plty = 1,
  grid.col = "grey", cglty = 1, cglwd = 0.8,
  axislabcol = "grey30",
  vlabels = colnames(radar_df),
  caxislabels = round(seq(from = 0, to = radar_df["max",1], length.out = 5)),
  title = cell_name )

legend( x = "bottomleft", legend = rownames(radar_df) - c("max", "min", lab_c1, lab_c2),
  horiz = FALSE, bty = "n", pch = 20, col = color, text.col = "black", cex = 1, pt.cex = 1.5 )

Get radar df of relative numbers of interactions for a certain cell type

Description

# Get radar plot of relative numbers of interactions for a certain cell type
# @param tab_c1 barplot dataframe from Viewpoint generated by getBarplotDF2() containing data for condition 1
# @param tab_c2 barplot dataframe from Viewpoint generated by getBarplotDF2() containing data for condition 2
# @param tab_c3 barplot dataframe from Viewpoint generated by getBarplotDF2() containing data for condition 3
# @param lab_c1 label for condition 1
# @param lab_c2 label for condition 2
# @param lab_c3 label for condition 3
# @param cell_name label of cell type of interest
# @return plot
@importFrom fmsb radarchart
@importFrom data.table transpose
getRadarPlot <- function(tab_c1, tab_c2, tab_c3, lab_c1, lab_c2, lab_c3, cell_name)
if(is.null(tab_c3))
  df <- merge(tab_c1, tab_c2, by = "Clusters", all = TRUE)
  colnames(df) <- c("Clusters", "nint_c1", "nint_c2")
else
  df <- merge(tab_c1, tab_c2, by = "Clusters", all = TRUE)
  df <- merge(df, tab_c3, by = "Clusters", all = TRUE)
  colnames(df) <- c("Clusters", "nint_c1", "nint_c2", "nint_c3")

  if(is.null(lab_c3))
    rownames(radar_df) <- c("max", "min", lab_c1, lab_c2)
  else
    rownames(radar_df) <- c("max", "min", lab_c1, lab_c2, lab_c3)

colnames(radar_df) <- cluster_names

color <- c("#438ECC", "#E97778", "#00BA38")

fmsb::radarchart( radar_df, axistype = 1,
  polygon.pcol = color, pfcol = scales::alpha(color, 0.5),
  plwd = 2, plty = 1,
  grid.col = "grey", cglty = 1, cglwd = 0.8,
  axislabcol = "grey30",
  vlabels = colnames(radar_df),
  caxislabels = round(seq(from = 0, to = radar_df["max",1], length.out = 5)),
  title = cell_name )

legend( x = "bottomleft", legend = rownames(radar_df) - c("max", "min", lab_c1, lab_c2),
  horiz = FALSE, bty = "n", pch = 20, col = color, text.col = "black", cex = 1, pt.cex = 1.5 )

Get radar df of relative numbers of interactions for a certain cell type
getRadar_df

"nint_c1", "nint_c2") else df <- merge(tab_c1, tab_c2, by = "Clusters", all = TRUE) df <- merge(df, tab_c3, by = "Clusters", all = TRUE) colnames(df) <- c("Clusters", "nint_c1", "nint_c2", "nint_c3")

df[is.na(df)] <- 0

cluster_names <- df$Clusters # add max and min
max_nint <- max(df[, -1]) df <- add_column(df, max_nint, after = "Clusters") df <- add_column(df, "min_nint" = 0, after = "max_nint")

radar_df <- data.table::transpose(df[, -1])

if(is.null(lab_c3)) rownames(radar_df) <- c("max", "min", lab_c1, lab_c2) else rownames(radar_df) <- c("max", "min", lab_c1, lab_c2, lab_c3)

colnames(radar_df) <- cluster_names

color <- c("#438ECC", "#E97778", "#00BA38")

fmsb::radarchart(radar_df, axistype = 1, # Customize the polygon pcol = color, pcoll = scales::alpha(color, 0.5), plwd = 2, plty = 1, # Customize the grid clgcol = "grey", clgty = 1, clgwd = 0.8, # Customize the axis axislabcol = "grey30", # Variable labels vlcex = 1.2, vlables = colnames(radar_df), caxislabels = round(seq(from = 0, to = radar_df["max",1], length.out = 5)), title = cell_name ) legend( x = "bottomleft", legend = rownames(radar_df[-c(1,2),]), horiz = FALSE, bty = "n", pch = 20 , col = color, text.col = "black", cex = 1, pt.cex = 1.5 )

Get radar df of relative numbers of interactions for a certain cell type

Usage

getRadar_df(tab_c1, tab_c2, tab_c3, lab_c1, lab_c2, lab_c3)

Arguments

tab_c1 barplot dataframe from Viewpoint generated by getBarplotDF2() containing data for condition 1

tab_c2 barplot dataframe from Viewpoint generated by getBarplotDF2() containing data for condition 2

tab_c3 barplot dataframe from Viewpoint generated by getBarplotDF2() containing data for condition 3

lab_c1 label for condition 1

lab_c2 label for condition 2

lab_c3 label for condition 3

Value

df to be then used with fmsb radarchart
getRankedTerms

Get table with ranked functional terms

Description
Get table with ranked functional terms

Usage
getRankedTerms(data.fun.annot)

Arguments

- data.fun.annot: annotated df (GO/path/combined)

Value
table with ranking

getSignificantFunctions

Calculate significant function per intpair module

Description
Calculate significant function per intpair module

Usage
getSignificantFunctions(
    subGenePairs_func_mat,
    gpModules_assign,
    rank.terms,
    input_maxPval
)

Arguments

- subGenePairs_func_mat: subset of binary mat
- gpModules_assign: assignment of intpairs to modules
- rank.terms: table of ranked functions
- input_maxPval: threshold of significance
getSignificantFunctions_multiCond

Get significance of functional terms related to unique int-pairs per condition

Value

table with significant functions

Usage

getSignificantFunctions_multiCond(sub_annot, unique_intpairs)

Arguments

sub_annot  annotation matrix subset to unique int-pairs
unique_intpairs  data.frame with unique int-pairs by condition

Value

data.frame with calculated pvalue of significance

getSignif_table  Wrapper for other functions to get significant table of func terms

Description

Wrapper for other functions to get significant table of func terms

Usage

getSignif_table(
  data_cond1,
  data_cond2,
  data_cond3,
  lab_c1,
  lab_c2,
  lab_c3,
  annot_cond1,
  annot_cond2,
  annot_cond3
)
Arguments

- **data_cond1**: filt.data() corresponding to chosen condition 1
- **data_cond2**: filt.data() corresponding to chosen condition 2
- **data_cond3**: filt.data() corresponding to chosen condition 3
- **lab_c1**: data label for condition 1
- **lab_c2**: data label for condition 2
- **lab_c3**: data label for condition 3
- **annot_cond1**: binary matrix int-pair by functions for cond1
- **annot_cond2**: binary matrix int-pair by functions for cond2
- **annot_cond3**: binary matrix int-pair by functions for cond3

Value

- list containing pvalue_df and unique_intpairs df

---

**getSunburst**

*Get Sunburst plot of selected functional terms*

Description

Get Sunburst plot of selected functional terms

Usage

```r
getSunburst(
  sel.data,
  func_selected,
  int_p_fun,
  cluster.colors,
  input_num_or_weight_radio
)
```

Arguments

- **sel.data**: dataframe of selected functions
- **func_selected**: the selected functional term
- **int_p_fun**: dataframe with int pairs annotated to this function
- **cluster.colors**: for plotting
- **input_num_or_weight_radio**: either num of interactions or weighted by score

Value

- plotly figure
getUMAPipModules

Get UMAP for IP modules

**Description**

Get UMAP for IP modules

**Usage**

getUMAPipModules(intPairs.dendro, gpModules_assign, ipm_colors)

**Arguments**

- **intPairs.dendro**
  - list output of dendrogram
- **gpModules_assign**
  - named vector of module assignment
- **ipm_colors**
  - for intpair modules

**Value**

plotly umap

getUniqueDotplot

Plot dotplot containing only unique int-pair/cluster pairs with many conditions

**Description**

Plot dotplot containing only unique int-pair/cluster pairs with many conditions

**Usage**

getUniqueDotplot(data_dotplot, clust.order)

**Arguments**

- **data_dotplot**
  - table with selected int_pairs for multiple conditions
- **clust.order**
  - how to order clusters

**Value**

ggplot object
**getUniqueIntpairs_byCond**

*Get table of unique int-pairs by condition*

**Description**

Get table of unique int-pairs by condition

**Usage**

```r
getUniqueIntpairs_byCond(
  data_cond1,
  data_cond2,
  data_cond3 = NULL,
  lab_c1,
  lab_c2,
  lab_c3 = NULL
)
```

**Arguments**

- `data_cond1`: filt.data() corresponding to chosen condition 1
- `data_cond2`: filt.data() corresponding to chosen condition 2
- `data_cond3`: filt.data() corresponding to chosen condition 3
- `lab_c1`: data label for condition 1
- `lab_c2`: data label for condition 2
- `lab_c3`: data label for condition 3

**Value**

modified merged filt.data containing only unique intpairs

---

**goLink**

*Get GO link*

**Description**

Get GO link

**Usage**

```r
goLink(go_id)
```
**Arguments**

- **go_id** string

**Value**

html link to website

---

**Description**

A dataset obtained from Tirosh et al melanoma dataset, running CellPhoneDBv2. This data is generated by InterCellar running read.CPDBv2()

**Usage**

input.data

**Format**

A data frame with 5638 rows and 11 variables:

- **int_pair** interaction pair name, geneA & geneB
- **geneA** name, hgnc_symbol
- **geneB** name, hgnc_symbol
- **typeA** molecular type of geneA, either L (ligand) or R (receptor)
- **typeB** molecular type of geneB, either L (ligand) or R (receptor)
- **clustA** name of first cluster, either character or number
- **clustB** name of second cluster, either character or number
- **score** int-pair score as avg expression of geneA and geneB over clustA and clustB, decimal
- **p_value** int-pair pvalue, decimal
- **annotation_strategy** database from which the int-pair was retrieved
- **int.type** either autocrine or paracrine
### read.cellchat

**Description**

Read dataframe of cell-cell communication from CellChat (ligand/receptor)

**Usage**

```r
define_cellchat(file_tab)
```

**Arguments**

- `file_tab` dataframe from cellchat

**Value**

input.data formatted for InterCellar

### read.CPDBv2

**Description**

Read output from CellPhoneDB v2.

Output is a folder containing 4 .txt files:
- deconvoluted.txt: containing list of single genes and their mean expression in each cluster (not considered);
- means.txt: containing list of interacting pairs with info regarding L/R, annotation strategy and mean value of all pairs over cluster couples.
- pvalues.txt: same as means, but containing pvalue of each pair, for each cluster couple.
- significant_means.txt: only means of those pairs that have pvalue < 0.05. Has one more column: rank. If the statistical analysis is not run, the folder would contain only deconvoluted and means.

**Usage**

```r
read.CPDBv2(folder)
```

**Arguments**

- `folder` folder containing output

**Value**

input.data which is the pre-processed object with annotated L-R pairs
**read.customInput**  
*Read custom input file and re-structure it with InterCellar format*

**Description**
Read custom input file and re-structure it with InterCellar format

**Usage**
```
read.customInput(tab, separator)
```

**Arguments**
- **tab**: custom input table
- **separator**: character that separates two elements of an interaction pair

**Value**
preprocessed table

---

**read.icellnet**  
*Read ICELLNET dataframe*

**Description**
Read ICELLNET dataframe

**Usage**
```
read.icellnet(tab, input_icellnet_CC, input_icellnet_dir)
```

**Arguments**
- **tab**: dataframe with int-pairs in "X" column, other columns as cell types
- **input_icellnet_CC**: central cell name
- **input_icellnet_dir**: direction of interaction either out or in

**Value**
pre-processed input data
**Description**

SCSR description: the output folder is a collection of txt files, one for each clusters pair considered. The "paracrine" option looks for ligands expressed in cluster A and their associated receptors according to LRdb 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". This file is a 4-column table: ligands, receptors, interaction types ("paracrine", "autocrine", "autocrine/paracrine" and "specific"), and the associated LRscore. InterCellar: rename autocrine|paracrine to paracrine

**Usage**

```r
read.SCsignalR(folder)
```

**Arguments**

- `folder` containing output from SingleCellSignalR, named cell-signaling

**Value**

- `input.data`: preprocessed object with annotated L-R pairs

---

**run_app**

**Run the Shiny Application**

**Description**

Run the Shiny Application

**Usage**

```r
run_app(reproducible = TRUE)
```

**Arguments**

- `reproducible` boolean for setting a seed, making plots reproducible
Value

a running instance of InterCellar

Examples

## Not run:
run_app()

## End(Not run)

---

subsetAnnot_multiCond Subset int-pair by function matrices to unique int-pairs by condition

Description

Subset int-pair by function matrices to unique int-pairs by condition

Usage

subsetAnnot_multiCond(
  annot_cond1,
  annot_cond2,
  annot_cond3,
  unique_intpairs,
  lab_c1,
  lab_c2,
  lab_c3
)

Arguments

annot_cond1 binary matrix int-pair by functions for cond1
annot_cond2 binary matrix int-pair by functions for cond2
annot_cond3 binary matrix int-pair by functions for cond3
unique_intpairs table of unique int-pairs by condition
lab_c1 label cond1
lab_c2 label cond2
lab_c3 label cond3

Value

subset merged matrix
subsetFuncMatBYFlow  Subset pairs-function matrix by selected flow

Description
Subset pairs-function matrix by selected flow

Usage
subsetFuncMatBYFlow(pairs_func_matrix, flow_df)

Arguments
pairs_func_matrix  binary
flow_df  subset of input data by flow

Value
subset of binary mat

swap.RLint  Swaps interaction pairs that are R-L to L-R

Description
Swaps interaction pairs that are R-L to L-R

Usage
swap.RLint(RLint)

Arguments
RLint  subset of R-L interactions

Value
input data with ordered L-R pairs and L-L/R-R
### uniprotLink

**Get html link to uniprot**

**Description**
Get html link to uniprot

**Usage**
uniprotLink(uniprot)

**Arguments**
- uniprot  symbol

**Value**
- html link to website

### updateInputLR

**Function that orders all interaction pairs as L-R. Leaves unchanged the R-R and L-L**

**Description**
Function that orders all interaction pairs as L-R. Leaves unchanged the R-R and L-L

**Usage**
updateInputLR(input.data)

**Arguments**
- input.data  uploaded data

**Value**
- ordered input data
Index

* datasets
  - input.data, 27

annotateGO, 3
annotatePathways, 4
buildPairsbyFunctionMatrix, 4
checkLL_RR, 5
circlePlot, 5
combineAnnotations, 6
createBarPlot1_ggplot, 6
createBarPlot2_CV, 7
createBarPlot2_ggplot, 7
createBarPlot_CV, 8
createNetwork, 9
dendroIntPairModules, 9
elbowPoint, 10
ensembLink, 10
getBack2BackBarplot, 11
getBarplotDF, 11
getBarplotDF2, 12
getClusterA_Names, 12
getClusterColors, 13
getClusterNames, 13
getClusterNetwork, 14
getClusterSize, 14
getDistinctCouplets, 15
getDotPlot_selInt, 15
geneGeneTable, 16
geneGObiomaRt, 17
geneHitsf, 17
geneIntFlow, 18
geneNtermsBYdb, 18
geneNumLR, 19
genePieChart, 19
geneRadar_df, 20
geneRankedTerms, 22
getSignif_table, 23
geneSignificantFunctions, 22
geneSignificantFunctions_multiCond, 23
geneSunburst, 24
geneUMAPipModules, 25
geneUniqueDotplot, 25
geneUniqueIntpairs_byCond, 26
goLink, 26
input.data, 27
read.cellchat, 28
read.CPDBv2, 28
read.customInput, 29
read.icellnet, 29
read.SCsignalR, 30
run_app, 30
subsetAnnot_multiCond, 31
subsetFuncMatBYFlow, 32
swap.RLink, 32
uniprotLink, 33
updateInputLR, 33