# Package 'InteRact'

July 5, 2018

**Title** InteRact : Analysis of proteomics data

**Version** 0.1.0

**Date** 2018-02-21

**Author** Guillaume Voisinne

**Maintainer** Guillaume Voisinne <voisinne@ciml.univ-mrs.fr>

**Description** The InteRact package provides tools to analyse affinity purification data

**Imports** dplyr,
ggplot2,
ggrepel,
stringr,
Hmisc,
igraph,
networkD3,
ggsignif,
PSICQUIC

**LazyData** TRUE

**License** file LICENSE

**RoxygenNote** 6.0.1

**Depends** R (>= 2.10)

## R topics documented:

---

`add_GO_data` *Add GO annotations*

---

## Description

Add GO annotations corresponding to a set of protein identifiers

## Usage

```
add_GO_data(df, map_id = "Entry", GO_type = "molecular_function",
  organism = "mouse", slim = FALSE, updateProgress = NULL)
```

### Arguments

| | |
|---|---|
| df | a data.frame with protein IDs in column map_id |
| map_id | column name used to map protein identifiers |
| GO_type | type of GO term ("molecular_function", "biological_process" or "cellular_component") |
| organism | organism for which the annotations have to be mapped |
| slim | logical, use GO_slim annotations |
| updateProgress | logical, function to show progress in shiny app |

### Value

a data.frame with an additional GO annotation column

---

## add_Hallmark_data *Add Hallmark annotations*

---

### Description

Add Hallmark annotations corresponding to a set of protein identifiers

### Usage

```
add_Hallmark_data(df, map_id = "Gene.names...primary..",
  updateProgress = NULL)
```

### Arguments

| | |
|---|---|
| df | a data.frame with protein IDs in column map_id |
| map_id | column name used to map protein identifiers |
| updateProgress | logical, function to show progress in shiny app |

### Value

a data.frame with an additional Hallmark annotation column

---

## add_KEGG_data *Add KEGG pathway annotations*

---

### Description

Add KEGG pathway annotations corresponding to a set of KEGG IDs

### Usage

```
add_KEGG_data(df, map_id = "Cross.reference..KEGG.", organism = "mouse",
  updateProgress = NULL)
```

### Arguments

| | |
|---|---|
| df | a data.frame with KEGG IDs in column map_id |
| map_id | column name used to map KEGG IDs |
| organism | organism for which the annotations have to be mapped |
| updateProgress | logical, function to show progress in shiny app |

### Value

a data.frame with an additional KEGG pathway annotation column

---

## analyse_interactome *Construct an interactome by comparing bait and control background across experimental conditions*

---

### Description

Construct an interactome by comparing bait and control background across experimental conditions

### Usage

```
analyse_interactome(df, ibait, bait_gene_name, Npep, Protein.IDs, name_bait,
  name_ctrl, background, conditions, replicates, by_conditions = TRUE,
  pool_background = TRUE, log_test = TRUE, log_stoichio = TRUE,
  substract_ctrl = TRUE)
```

### Arguments

| | |
|---|---|
| df | a data frame of protein intensities. columns are experimental samples and rows are proteins |
| ibait | : row index corresponding to the bait protein |
| bait_gene_name | : The gene name of the bait |
| Npep | : vector containing the number of theoretically observable peptide per protein (same length as dim(df)[1]) |
| Protein.IDs | : vector containing protein IDs (same length as dim(df)[1]) |
| name_bait | : name of the bait as appearing in the background vector |
| name_ctrl | : name of the control as appearing in the background vector |
| background | : vector of background names for each experimental sample |
| conditions | : vector of conditions for each each experimental sample |
| replicates | : vector of biological replicates for each each experimental sample |
| by_conditions | option to perform the comparison between bait and control group for each condition |
| pool_background | option to use all control background conditions as one control group for all conditions |
| log_test | logical, perform t-test on log transform intensities |
| log_stoichio | logical, use the geometric mean instead of the arithmetic mean to compute stoichiometries |
| substract_ctrl | logical, substract ctrl intensities in the calculation of stoichiometries |

## Value

an object of class `InteRactome`, i.e a list including the following elements :

`conditions` : a vector of experimental conditions.

`names` : a vector of names (by default gene names are used).

`p_val` : a list of vectors containing the p values associated to each experimental condition.

`fold_change` : a list of vectors containing the fold change associated to each experimental condition.

`...` : other variables.

---

`annotation_enrichment_analysis`
*Perform enrichment analysis*

---

## Description

Perform enrichment analysis for protein annotations stored in a formatted data.frame

## Usage

```
annotation_enrichment_analysis(df, idx_detect,
  annotation_selected = c("Keywords", "Protein.families"),
  names = df$Gene.names...primary.., organism = "mouse",
  updateProgress = NULL, showProgress = TRUE, orderOutput = TRUE)
```

## Arguments

| | |
|---|---|
| `df` | a formatted data.frame with annoations corresponding to each row. Types of annotations are organized by columns. |
| `idx_detect` | indexes of the foreground set. |
| `annotation_selected` | set of annotations on which to perform the analysis |
| `names` | row names. Used in the output data.frame |
| `organism` | organism for which the analysis is to be performed ("mouse" or "human") |
| `updateProgress` | logical, function to show progress in shiny app |
| `showProgress` | logical, show progress in console |
| `orderOutput` | logical, order annotations by enrichment p-values in the output data.frame |

## Value

a data.frame

## append_annotations *Append annotations to an* InteRactome

### Description

Append annotations to an InteRactome

### Usage

```
append_annotations(res, annotations = NULL, name_id = "Protein.IDs",
  organism = "mouse")
```

### Arguments

| | |
|---|---|
| res | an InteRactome |
| annotations | type of annotations to append |
| name_id | column name used to map protein identifiers |
| organism | organism for which the annotations have to be appended |

### Value

an InteRactome

## append_FDR *Append a FDR column to an* InteRactome

### Description

Append a FDR column to an InteRactome

### Usage

```
append_FDR(res, df)
```

### Arguments

| | |
|---|---|
| res | an InteRactome |
| df | a data.frame containing (at least) columns 'bait', names', 'FDR' and 'conditions' |

### Value

an InteRactome

## Examples

```
# #load data :
data("proteinGroups_Cbl")
#Run InteRact with default parameters
res <- InteRact(proteinGroups_Cbl, bait_gene_name = "Cbl")$Interactome
df_merge <- merge_conditions(res)
df_FDR <- compute_FDR_from_asymmetry(df_merge)
Interactome <- append_FDR(res, df_FDR)
```

---

# append_PPI *Append protein-protein interaction*

---

## Description

Append protein-protein interaction information to an `InteRactome`. PPI are retrieved from databases IntAct, MINT, BioGRID and HPRD

## Usage

```
append_PPI(res, mapping = "names")
```

## Arguments

| | |
|---|---|
| `res` | an `InteRactome` |
| `mapping` | name of the `InteRactome`'s variable containing gene names |

## Value

an `InteRactome`

---

# average_technical_replicates *Average protein intensities over technical replicates*

---

## Description

Average protein intensities over technical replicates

## Usage

```
average_technical_replicates(df, cond, log = TRUE)
```

## Arguments

| | |
|---|---|
| `df` | A data frame of protein intensities |
| `cond` | A data frame containing the description of df's columns (i.e "idx", bckg", "time", "bio" and "tech") as returned by function `identify_conditions()` |
| `log` | use geometric mean |

## Value

A list containing :

`Intensity`, a data frame of protein intensities averaged over technical replicates;

`conditions`, a data frame containing the description of `Intensity`'s columns

## Examples

```
#load data :
data("proteinGroups_Cbl")
cond <- identify_conditions(proteinGroups_Cbl)
Column_intensity_pattern <- "^Intensity."
df_int <- proteinGroups_Cbl[ , grep(Column_intensity_pattern, colnames(df))]
```

---

# compute_correlations *Compute correlation in protein recruitment*

---

## Description

Compute correlation in protein recruitment from interaction stoichiometries computed across all conditions for each biological replicate. Pearson correlations are used.

## Usage

```
compute_correlations(res, idx = NULL)
```

## Arguments

| | |
|---|---|
| `res` | an `InteRactome` |
| `idx` | indexes of the set of proteins for which correlations will be computed |

## Value

a data.frame with protein correlation information (correlation coefficient in column 'r_corr' and associated p-value in column 'p-corr')

---

# compute_FDR_from_asymmetry *Compute the FDR from the asymmetry of the volcano plot*

---

## Description

Compute the FDR (False Discovery Rate) using the asymmetry of the volcano plot. It uses the fonction f(x) = c / (x-x0) with x = log10(fold_change), y=-log10(p_value). Points with x>x0 and y>f(x) are taken as true positive (TP) Points with x<x0 and y>f(x) are taken as false positive (FP) For a given set of parameters (c,x0), the FDR is given by TP/(TP+FP)

## Usage

```
compute_FDR_from_asymmetry(df, c = seq(from = 0, to = 4, by = 0.1),
  x0 = seq(from = 0, to = 3, by = 0.1))
```

### Arguments

| | |
|---|---|
| df | : a data.frame containing columns p_val and fold_change |
| c | : numeric vector |
| x0 | : numeric vector |

### Value

a data.frame with a extra column FDR.

If parameters c and x0 are vectors, FDR is taken as the minimum FDR value across all sets of parameters

### Examples

```
# #load data :
data("proteinGroups_Cbl")
#Run InteRact with default parameters
res <- InteRact(proteinGroups_Cbl, bait_gene_name = "Cbl")$Interactome
df_merge <- merge_conditions(res)
df_FDR <- compute_FDR_from_asymmetry(df_merge)
Interactome <- append_FDR(res, df_FDR)
```

---

## create_summary_table_PPI

*Retrieve protein-protein interaction information from databses IntAct, MINT, BioGRID and HPRD*

---

### Description

Retrieve protein-protein interaction information from databses IntAct, MINT, BioGRID and HPRD

### Usage

```
create_summary_table_PPI(gene_name)
```

### Arguments

| | |
|---|---|
| gene_name | the gene name for which to retrieve PPI |

### Value

a data.frame PPI information

## discretize_values *Discretize values in a vector based on a finite set of values*

### Description

Discretize values in a vector based on a finite set of values

### Usage

```
discretize_values(x, breaks = c(1, 0.1, 0.05, 0.01),
  decreasing_order = TRUE)
```

### Arguments

| | |
|---|---|
| x | numeric vector |
| breaks | numeric vector. Set of discrete values on which x values will be mapped. Non-mapped values will be set to NA |
| decreasing_order | logical. Map beaks values from the greatest to the smallest |

### Value

a numeric vector

## dot_plot *Dot plot representation of matrices*

### Description

Dot plot representation of matrices

### Usage

```
dot_plot(Dot_Size, Dot_Color = NULL, title = "Dot Plot",
  size_range = range(Dot_Size), size_var = "size", color_var = "color")
```

### Arguments

| | |
|---|---|
| Dot_Size | a matrix of dot sizes |
| Dot_Color | a matrix of dot colors (optionnal) |
| title | plot title |
| size_range | range of dot sizes to display |
| size_var | name of the variable corresponding to dot size |
| color_var | name of the variable corresponding to dot color |

### Value

a plot

## estimate_Npep *Get the number of theoretically observable peptides per protein*

### Description

Get the number of theoretically observable peptides per protein

### Usage

```
estimate_Npep(df, Column_Npep = NULL)
```

### Arguments

| | |
|---|---|
| df | A data frame |
| Column_Npep | column containing the number of theoretically observable peptides per protein. If NULL try to compute the number of theoretically observable peptides using iBAQ values, or use molecular weight. |

### Value

A data frame with the column 'Npep'

## filter_conditions *Filter conditions from an interactome*

### Description

Filter conditions from an interactome

### Usage

```
filter_conditions(res, conditions_to_filter_out)
```

### Arguments

| | |
|---|---|
| res | an InteRactome |
| conditions_to_filter_out | character vector with names of conditions to filter out |

### Value

an InteRactome

## filter_Proteins — *Filtering of a data frame using a threshold on protein identification score and gene names*

### Description

Filtering of a data frame using a threshold on protein identification score and gene names

### Usage

```
filter_Proteins(df, min_score = 0, Column_gene_name = "Gene.names",
  Column_score = "Score", split_param = ";")
```

### Arguments

| | |
|---|---|
| df | A data frame |
| min_score | Threshold for protein identification score |
| Column_gene_name | The name of df's column containing gene names |
| Column_score | The name of df's column containing protein identification score |
| split_param | Character used to split gene names into substrings. |

### Value

A filtered data frame. Contains an extra column with the first substring of the column Column_gene_name

## geom_mean — *Perform the geometric mean of a numeric vector*

### Description

Perform the geometric mean of a numeric vector

### Usage

```
geom_mean(x, na.rm = TRUE)
```

### Arguments

| | |
|---|---|
| x | A numeric vector |
| na.rm | remove NA values |

### Value

A numeric value

## get_annotations *Get annotations from uniprot for a set of protein identifiers*

### Description

Get annotations from uniprot for a set of protein identifiers. From a set of IDs, keep the first that correspond to a "reviewed" protein, or by default the first ID of the set name_id : column containing the set of protein identifiers separated by "split_param" organism = c("mouse", "human")

### Usage

```
get_annotations(data, name_id = "Protein.IDs", split_param = ";",
  organism = "mouse", updateProgress = NULL)
```

### Arguments

| | |
|---|---|
| data | a data.frame with protein IDs in column name_id |
| name_id | column name used to map protein identifiers |
| split_param | split character used to separate different protein IDs |
| organism | organism for which the annotations have to be appended |
| updateProgress | logical, function to show progress in shiny app |

### Value

an InteRactome

## get_PPI_from_BioGRID *Retrieve protein-protein interaction information from BioGRID*

### Description

Retrieve protein-protein interaction information from BioGRID

### Usage

```
get_PPI_from_BioGRID(gene_name, tax_ID = c(9606, 10090))
```

### Arguments

| | |
|---|---|
| gene_name | the gene name for which to retrieve PPI |
| tax_ID | taxon ID for which to retrieve PPI |

### Value

a data.frame PPI information

## get_PPI_from_HPRD *Retrieve protein-protein interaction information from HPRD*

### Description

Retrieve protein-protein interaction information from HPRD

### Usage

```
get_PPI_from_HPRD(gene_name)
```

### Arguments

| | |
|---|---|
| gene_name | the gene name for which to retrieve PPI |

## get_PPI_from_psicquic *Retrieve protein-protein interaction information using PSICQUIC*

### Description

Retrieve protein-protein interaction information using PSICQUIC

### Usage

```
get_PPI_from_psicquic(gene_name, tax_ID = c(9606, 10090),
  provider = c("IntAct", "MINT"))
```

### Arguments

| | |
|---|---|
| gene_name | the gene name for which to retrieve PPI |
| tax_ID | taxon ID for which to retrieve PPI |
| provider | database from which to retrieve PPI |

### Value

a data.frame PPI information

`global_analysis` *Adds global variables by analysing values across all conditions of an* `InteRactome`

## Description

Adds global variables by analysing values across all conditions of an `InteRactome`

## Usage

```
global_analysis(res)
```

## Arguments

| | |
|---|---|
| `res` | an `InteRactome` |

## Value

an `InteRactome` with global varaiables

`identify_conditions` *Identify conditions (background, time of stimulation, biological and technical replicates) from column names*

## Description

Identify conditions (background, time of stimulation, biological and technical replicates) from column names

## Usage

```
identify_conditions(df, Column_intensity_pattern = "^Intensity.",
  split = "_", bckg_pos = 1, bio_pos = 2, time_pos = 3, tech_pos = 4)
```

## Arguments

| | |
|---|---|
| `df` | A dataframe containing protein intensities. By default, protein intensity column names start by "Intensity." (use parameter `Column_intensity_pattern` to change) |
| `Column_intensity_pattern` | Pattern (regular exrpression) used to identfy df's columns containing protein intensity values |
| `split` | Character used to split column names into substrings |
| `bckg_pos` | Position of the sample background in splitted column names |
| `bio_pos` | Position of the sample biological replicate in splitted column names |
| `time_pos` | Position of the sample experimental condition in splitted column names |
| `tech_pos` | Position of the sample technical replicate in splitted column names |

## Value

a data frame describing experimental samples in terms of background, biological and technical replicates, and experimental conditions

## Examples

```
#load data :
data("proteinGroups_Cbl")
# You can identify columns and their description separately using \code{identify_conditions()}
cond <- identify_conditions(proteinGroups_Cbl)
print.data.frame(cond)
# and use it as parameters for function InteRact()
res <- InteRact(proteinGroups_Cbl, bait_gene_name = "Cbl", condition = cond)$Interactome
```

---

# identify_interactors *Identify specific interactors in an* InteRactome

---

## Description

Identify specific interactors in an InteRactome

## Usage

```
identify_interactors(res, var_p_val = "p_val", p_val_thresh = 0.05,
  fold_change_thresh = 2, n_success_min = 1, consecutive_success = FALSE,
  ...)
```

## Arguments

| | |
|---|---|
| res | an InteRactome |
| var_p_val | name of the p-value variable |
| p_val_thresh | p-value threshold |
| fold_change_thresh | fold-change threshold |
| n_success_min | minimal number of conditions in which the interactor must pass the the p-value and the fold-change thresholds |
| consecutive_success | logical, impose that the interactor must pass selection thresholds in n_success_min consecutive conditions. |
| ... | additionnal paramters passed to function order_interactome() |

## Value

an InteRactome with extra variables is_interactor, n_success and interactor

InteRact *Analysis of AP-MS data*

## Description

Analysis of AP-MS data

## Usage

```
InteRact(df, updateProgress = NULL, N_rep = 3, quantile_rep = 0.05,
  pool_background = TRUE, log_test = TRUE, log_stoichio = TRUE,
  log_mean = TRUE, by_conditions = TRUE, substract_ctrl = TRUE,
  preprocess_df = NULL, ...)
```

## Arguments

| | |
|---|---|
| df | A dataframe containing protein intensities. By default, protein intensity column names start by "Intensity." (use parameter Column_intensity_pattern to change) |
| updateProgress | function to show progress bar in shiny app |
| N_rep | Number of iterations for the replacement of missing values |
| quantile_rep | Numeric value between 0 and 1. Quantile of the distribution of mean intensities in the control background used to replace missing values. |
| pool_background | option to use all control background conditions as one control group for all conditions |
| log_test | logical, perform t-test on log transform intensities |
| log_stoichio | logical, use the geometric mean instead of the arithmetic mean to compute stoichiometries |
| log_mean | logical, use the geometric mean instead of the arithmetic mean to compute the mean InteRactome |
| by_conditions | option to perform the comparison between bait and control group for each condition |
| substract_ctrl | logical, substract ctrl intensities in the calculation of stoichiometries |
| preprocess_df | list obtained by the function preprocess_data() |
| ... | Additional parameters passed to function preprocess_data() and identify_conditions |

## Value

a list containing the preprocessed data and on object of class InteRactome, i.e a list including the following elements :

conditions : a vector of experimental conditions.

names : a vector of names (by default gene names are used).

p_val : a list of vectors containing the p values associated to each experimental condition.

fold_change : a list of vectors containing the fold change associated to each experimental condition.

... : other variables.

## Author(s)

Guillaume Voisinne

## Examples

```
#load data :
data("proteinGroups_Cbl")
#Run InteRact with default parameters
res <- InteRact(proteinGroups_Cbl, bait_gene_name = "Cbl")$Interactome

#You now have an \code{InteRactome}. See its elements.
class(res)
names(res)
#Generate volcano plots
plot_volcanos(res)

#Identify specific interactors
res <- identify_interactors(res, p_val_thresh = 0.05, fold_change_thresh = 2)

#Visualize interaction kinetics
plot_per_condition(res)

# Append protein abundance information
res <- merge_proteome(res)
# Append annotations
annot <- get_annotations(res)
res <- append_annotations(res,  annot)
#Create a summary data frame
sum_tbl <- summary_table(res)
```

---

# mean_analysis *Compute the mean* InteRactome *(on variables 'p_val', 'fold_cahnge', 'stoichio' and 'stoichio_bio') from a list of* InteRactomes

---

## Description

Compute the mean InteRactome (on variables 'p_val', 'fold_cahnge', 'stoichio' and 'stoichio_bio') from a list of InteRactomes

## Usage

```
mean_analysis(res, log = TRUE, na.rm = TRUE)
```

## Arguments

| | |
|---|---|
| res | a list of InteRactome |
| log | logical, use the geometric mean instead of the arithmetic mean |
| na.rm | logical, remove NA values |

## merge_conditions *Merge different conditions from different interactomes into a single data.frame*

### Description

Merge different conditions from different interactomes into a single data.frame

### Usage

```
merge_conditions(res, selected_conditions = NULL)
```

### Arguments

| | |
|---|---|
| res | a list of InteRactomes |
| selected_conditions | a character vector containing names of conditions to merge |

### Value

a data.frame with columns bait, names, Protein.IDs, conditions, p_val, fold_change

## merge_duplicate_groups *Merge protein groups with the same gene name.*

### Description

Merge protein groups with the same gene name.

### Usage

```
merge_duplicate_groups(df, idx_col = NULL, merge_column = "gene_name")
```

### Arguments

| | |
|---|---|
| df | A data frame |
| idx_col | idx of columns for which values will be merged across protein groups |
| merge_column | column to identify rows to be be merged |

### Value

A merged data frame

## merge_proteome *Add protein abundance to an* InteRactome

### Description

Add protein abundance to an InteRactome. Protein abundance are obtained from CD4+ effector T cells.

### Usage

```
merge_proteome(res)
```

### Arguments

| | |
|---|---|
| res | an InteRactome |

## moving_average *Performs a running average on a numeric vector*

### Description

Performs a running average on a numeric vector

### Usage

```
moving_average(x, n)
```

### Arguments

| | |
|---|---|
| x | a numeric vector |
| n | integer, radius of the moving avergae (number of points extending on each side of the center point on which the average is computed) |

### Value

a smoothed numeric vector

## order_interactome *Order proteins within an* InteRactome

### Description

Order proteins within an InteRactome

### Usage

```
order_interactome(res, var_p_val = "min_p_val", p_val_breaks = c(1, 0.1,
  0.05, 0.01))
```

### Arguments

| | |
|---|---|
| res | an InteRactome |
| var_p_val | name of the p-value variable |
| p_val_breaks | numeric vector to discretize p-value |

### Value

an InteRactome

## plot_2D_stoichio *Plot abundance versus interaction stoichiometries*

### Description

Plot abundance versus interaction stoichiometries

### Usage

```
plot_2D_stoichio(res, condition = "max", xlim = NULL, ylim = NULL,
  N_display = 30)
```

### Arguments

| | |
|---|---|
| res | an InteRactome |
| condition | condition selected. If "max", the maximum stoichiometry across conditions will be used. |
| xlim | range of x values |
| ylim | range of y values |
| N_display | maximum number of protein to display |

### Value

a plot

## plot_annotation_results *Plot the result of the annotation enrichment analysis*

### Description

Plot the result of the annotation enrichment analysis

### Usage

```
plot_annotation_results(df, p_val_max = 0.05, method_adjust_p_val = "fdr",
  fold_change_min = 2, N_annot_min = 2)
```

### Arguments

| | |
|---|---|
| df | a formatted data.frame obtained by the function annotation_enrichment_analysis() |
| p_val_max | threshold for the enrichment p-value |
| method_adjust_p_val | method to adjust p-value for multiple comparisons |
| fold_change_min | threshold for the enrichment fold-change |
| N_annot_min | minimum number of elements that are annotated in the foreground set |

### Value

a plot

## plot_comparison *Plot protein intensities per biological replicate and background*

### Description

Plot protein intensities per biological replicate and background

### Usage

```
plot_comparison(res, name, conditions, textsize = 3, test = "t.test",
  test.args = list(paired = FALSE), map_signif_level = c(*** = 0.001, **
  = 0.01, * = 0.05), position = "position_jitter",
  position.args = list(width = 0.3, height = 0))
```

### Arguments

| | |
|---|---|
| res | an InteRactome |
| name | name of the protein to display |
| conditions | set of conditions to display |
| textsize | size of labels corresponding to significance levels |
| test | name of the test function to compare intensities between background |
| test.args | arguments passed to function test |
| map_signif_level | named vector with labels and corresponding significance levels |
| position | name of the function used to position data points |
| position.args | arguments passed to function position |

### Value

a plot

---

## plot_correlation_network

*Plot an interactive correlation network with communities highlighted*

---

### Description

Plot an interactive correlation network with communities highlighted

### Usage

```
plot_correlation_network(df_corr, r_corr_thresh = 0.8, p_val_thresh = 0.05)
```

### Arguments

| | |
|---|---|
| df_corr | a data.frame with columns 'r_corr' and 'p_corr' |
| r_corr_thresh | threshold for variable 'r_corr' (min) |
| p_val_thresh | threshold for ariable 'p_corr' (max) |

### Value

an interactive networkD3 plot

plot_per_condition *Dot plot representation of interaction as a function of experimental conditions*

## Description

Dot plot representation of interaction as a function of experimental conditions

## Usage

```
plot_per_condition(res, idx_cols = 1:length(res$conditions),
  idx_rows = 1:20, size_var = "norm_stoichio", size_range = c(0, 1),
  color_var = "p_val", color_breaks = c(1, 0.1, 0.05, 0.01),
  color_default = 1, save_file = NULL, plot_width = 2.5 +
  length(res$conditions)/5, plot_height = 2 + length(idx_rows)/5,
  clustering = TRUE)
```

## Arguments

| | |
|---|---|
| res | an InteRactome |
| idx_cols | numeric vector to select and order conditions to be displayed |
| idx_rows | numeric vector to select proteins to display |
| size_var | name of the variable corresponding to dot size |
| size_range | range of dot sizes to display |
| color_var | name of the variable corresponding to dot color |
| color_breaks | vector used to discretize colors |
| color_default | value corresponding to the default color |
| save_file | path of output file (.pdf) |
| plot_width | width of the output .pdf file |
| plot_height | height of the output .pdf file |
| clustering | logical or numeric vector. If logical, use hierarchical clustering to order proteins. If numeric, ordering indexes for displayed proteins (must be the same length as idx_rows) |

## Value

a list conataining :

a plot "plot"

a numeric vector "idx_order" containing the position of the protein displayed within the InteRactome

## plot_QC *Quality check plots for preprocessed AP-MS data*

### Description

Quality check plots for preprocessed AP-MS data

### Usage

```
plot_QC(prep_data)
```

### Arguments

| | |
|---|---|
| prep_data | preprocessed data as obtained using function preprocess_data() |

### Value

Several QC plots

## plot_stoichio *Plot interaction stoichiometries per biological replicate*

### Description

Plot interaction stoichiometries per biological replicate

### Usage

```
plot_stoichio(res, name, conditions = res$conditions,
  ref_condition = res$conditions[1], test = "t.test",
  test.args = list(paired = TRUE), map_signif_level = c(*** = 0.001, **
  = 0.01, * = 0.05), save_file = NULL)
```

### Arguments

| | |
|---|---|
| res | an InteRactome |
| name | name of the protein to display |
| conditions | set of conditions to display |
| ref_condition | name of the reference condition for all test |
| test | name of the test function to compare sttoichiometries between conditions |
| test.args | arguments passed to function test |
| map_signif_level | named vector with labels and corresponding significance levels |
| save_file | path of output file (.pdf) |

### Value

a plot

## plot_volcanos *Plot protein enrichement fold-change versus p-value*

### Description

Plot protein enrichement fold-change versus p-value

### Usage

```
plot_volcanos(res, labels = NULL, N_print = 15, conditions = NULL,
  p_val_thresh = 0.05, fold_change_thresh = 2, save_file = NULL,
  xlim = NULL, ylim = NULL, asinh_transform = TRUE)
```

### Arguments

| | |
|---|---|
| res | an InteRactome |
| labels | labels for proteins in plot. Must the same length as res$names |
| N_print | maximum of protein labels to display |
| conditions | conditions to plot |
| p_val_thresh | threshold on p-value to display |
| fold_change_thresh | threshold on fold-change to display |
| save_file | path of output file (.pdf) |
| xlim | range of x values |
| ylim | range of y values |
| asinh_transform | logical, display asinh(log10(p-value)) on the y-axis |

### Value

a plot

## preprocess_data *Preprocessing of raw data*

### Description

Preprocessing of raw data

### Usage

```
preprocess_data(df, Column_gene_name = "Gene.names", Column_score = "Score",
  Column_ID = "Protein.IDs", Column_Npep = NULL,
  Column_intensity_pattern = "^Intensity.", bait_gene_name,
  condition = NULL, bckg_bait = bait_gene_name, bckg_ctrl = "WT",
  log = TRUE, filter_time = NULL, filter_bio = NULL, filter_tech = NULL,
  ...)
```

### Arguments

| | |
|---|---|
| df | Data.frame with protein intensities |
| Column_gene_name | Column with gene names |
| Column_score | Column with protein identification score |
| Column_ID | Column with protein IDs |
| Column_Npep | Column with number of theoretically observable peptides per protein |
| Column_intensity_pattern | Pattern (regular exrpression) used to identfy df's columns containing protein intensity values |
| bait_gene_name | The gene name of the bait |
| condition | data.frame with columns "sample", bckg", "bio", "time" and "tech" indicating for each intensity column ("sample") its corresponding background ("bckg"), biologicla replicate ("bio), experimental condition ("tine) and technical replicate ("tech). |
| bckg_bait | Name of the bait background as found in condition$bckg (see below) |
| bckg_ctrl | Name of the control background as found in condition$bckg (see below) |
| log | logical, use geometric mean to average technical replicates |
| filter_time | vector of experimental conditions to exclude from analysis |
| filter_bio | vector of biological replicates to exclude from analysis |
| filter_tech | vector of technical replicates to exclude from analysis |
| ... | Additional parameters passed to function identify_conditions |

---

## proteinGroups_Cbl *Characterization of protein groups identified from AP-MS data using MaxQuant*

---

### Description

A dataset containing the characterization of protein groups identified using MaxQuant from AP-MS samples. Samples correspond to pull-down experiments performed on Cd4+ T cells from CBL-OST mice (expressing a tagged version of the bait protein Cbl) and WT mice (expressing the endogeneous Cbl protein). Cd4+ T cells from CBL-OST and WT backgrounds were stimulated by cross-linking of Cd3e and Cd4 for different times (0, 30, 120, 300 and 600 seconds). Samples were then lysed and subjected to MS-MS analysis. Experiments were conducted in three biological replicates (S1, S2 and S3). Sample were analysed by MS-MS in three technical replicates (R1, R2 and R3).

### Usage

```
proteinGroups_Cbl
```

### Format

A data frame with 1978 rows (protein groups) and 849 variables:

See the Maxquant documentation for the description of the different variables

## rescale_median *Normalize data frame by columns using the median*

### Description

Normalize data frame by columns using the median

### Usage

```
rescale_median(df)
```

### Arguments

| | |
|---|---|
| df | A data frame |

### Value

A normalized data frame

## row_mean *Compute the mean by row*

### Description

Compute the mean by row

### Usage

```
row_mean(df, na.rm = TRUE, log = FALSE)
```

### Arguments

| | |
|---|---|
| df | a data frame |
| na.rm | logical, remove NA values |
| log | logical, use geometric mean instead of arithmetic mean |

### Value

A numeric vector

## row_sd — *Compute the standard deviation by row*

### Description

Compute the standard deviation by row

### Usage

```
row_sd(df)
```

### Arguments

| | |
|---|---|
| df | a data frame |

### Value

A numeric vector

## row_stoichio — *Compute the stoichiometry of interaction using the method described in ...*

### Description

Compute the stoichiometry of interaction using the method described in ...

### Usage

```
row_stoichio(df, idx_group_1, idx_group_2, idx_bait, Npep, log = TRUE,
  substract_ctrl = TRUE)
```

### Arguments

| | |
|---|---|
| df | a data frame |
| idx_group_1 | column indexes corresponding to the first group (bait background) |
| idx_group_2 | column indexes corresponding to the second group (ctrl background) |
| idx_bait | row index for the bait protein |
| Npep | numeric vector containing the number of theoretically observable peptides for each protein |
| log | logical, use the geometric mean instead of the arithmetic mean |
| substract_ctrl | logical, substract ctrl intensities in the calculation of stoichiometries |

### Value

A numeric vector of interaction stoichiometries

## row_ttest *Perform a t-test comparison between two groups by row*

### Description

Perform a t-test comparison between two groups by row

### Usage

```
row_ttest(df, idx_group_1, idx_group_2, log = TRUE)
```

### Arguments

| | |
|---|---|
| df | a data frame |
| idx_group_1 | column indexes corresponding to the first group |
| idx_group_2 | column indexes corresponding to the second group |
| log | option to perform the t-test on log transformed data |

### Value

A data frame with columns 'p_val' and 'fold_change

## smooth_interactome *Smooth, using a moving average across conditions, selected variables of an* InteRactome

### Description

Smooth, using a moving average across conditions, selected variables of an InteRactome

### Usage

```
smooth_interactome(res, n = 1, order_conditions = NULL,
  var_smooth = c("fold_change", "p_val"))
```

### Arguments

| | |
|---|---|
| res | an InteRactome |
| n | integer, radius of the moving avergae (number of points extending on each side of the center point on which the average is computed) |
| order_conditions | a numeric vector ordering conditions in res$conditions |
| var_smooth | variables on which the moving average will be computed |

### Value

an smoothed InteRactome

summary_table *Create a summary table for an* InteRactome

## Description

Create a summary table for an InteRactome

## Usage

```
summary_table(res, add_columns = names(res))
```

## Arguments

| | |
|---|---|
| res | an InteRactome |
| add_columns | names of the variables to display in the summary table |

## Value

a data.frame

# Index