

# Package ‘XINA’

October 16, 2019

**Type** Package

**Title** Multiplexes isobaric mass tagged-based kinetics data for network analysis

**Version** 1.2.0

**Date** 2018-08-23

**Author** Lee, Lang Ho and Singh, Sasha A.

**Maintainer**

Lang Ho Lee <langholee@gmail.com> and Sasha A. Singh <sasingh@bwh.harvard.edu>

**Description** An intuitive R package simplifies network analyses output from multiplexed high-dimensional proteomics/transcriptomics kinetics data.

**Copyright** XINA combines multiple quantitative (kinetics) datasets from omics studies into a single input dataset for clustering.

Copyright(C)2018 Lang Ho Lee, Arda Halu, Stephanie Morgan, Hiroshi Iwata, Masanori Aikawa, and Sasha A. Singh This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program. If not, see <<https://www.gnu.org/licenses/>>. - Contact emails: L. Lee, LHLEE@BWH.HARVARD.EDU S. Singh, SASINGH@BWH.HARVARD.EDU M. Aikawa, MAIKAWA@BWH.HARVARD.EDU - Mailing address: Department of Medicine, Cardiovascular Division Center for Interdisciplinary Cardiovascular Sciences 3 Blackfan Street, 17th Floor Boston, MA 02115 USA

**Encoding** UTF-8

**LazyData** FALSE

**License** GPL-3

**Depends** R (>= 3.5), Biobase

**Imports** mclust, plyr, alluvial, ggplot2, igraph, gridExtra, tools, grDevices, graphics, utils, STRINGdb

**biocViews** ImmunoOncology, SystemsBiology, Proteomics, RNASeq, Network

**RoxygenNote** 6.1.0

**VignetteBuilder** knitr

**Suggests** knitr, rmarkdown

**git\_url** <https://git.bioconductor.org/packages/XINA>

**git\_branch** RELEASE\_3\_9

**git\_last\_commit** 78cf6c7

**git\_last\_commit\_date** 2019-05-02

**Date/Publication** 2019-10-15

## R topics documented:

add_legend . . . . .	3
alluvial_enriched . . . . .	3
alluvial_enrichment_tests . . . . .	5
calculate_centrality_scores . . . . .	6
default_size . . . . .	6
draw_alluvial_plot . . . . .	7
example_clusters . . . . .	8
extract_data_column . . . . .	8
find_similar_clusters . . . . .	9
generate_count_table . . . . .	9
generate_superset . . . . .	10
get_colors . . . . .	10
get_color_for_nodes . . . . .	11
get_comigrations_by_name . . . . .	11
get_condition_biased_comigrations . . . . .	12
get_layout . . . . .	13
get_mTOR_proteins . . . . .	14
get_random_data . . . . .	14
get_stats . . . . .	15
get_theme_blank . . . . .	15
get_unknown_ppi_nodes . . . . .	16
gn . . . . .	16
gn_desc . . . . .	17
hprd_ppi . . . . .	17
length2 . . . . .	17
load_previous_results . . . . .	18
make_random_xina_data . . . . .	19
mutate_colors . . . . .	20
organize_clusters . . . . .	21
plot_clusters . . . . .	21
plot_clusters_all . . . . .	22
plot_condition_compositions . . . . .	23
plot_enrichment_results . . . . .	24
plot_NA . . . . .	25
rank_centrality . . . . .	25
string_example . . . . .	26
xina_analysis . . . . .	26
xina_clustering . . . . .	27

*alluvial\_enriched* 3

xina_enrichment . . . . .	29
xina_plot_all . . . . .	30
xina_plot_bycluster . . . . .	32
xina_plot_single . . . . .	34
xina_result_example . . . . .	36

**Index** 37

add\_legend                    *add\_legend*

**Description**

Add plot legend and locate it outside of a network plot

**Usage**

```
add_legend(legend_location = "bottomright", ...)
```

**Arguments**

legend_location	Network centrality score matrix
...	Numeric, complex, or logical vectors.

**Value**

a legend to a plot

*alluvial\_enriched*            *alluvial\_enriched*

**Description**

'alluvial\_enriched' draws an alluvial plot and finds comigrated proteins. The comigration is a group of proteins that show the same expression pattern, classified and evaluated by XINA clustering, in at least two conditions. XINA can reduce the dataset complexity by filtering based on the number of comigrated proteins (size, 'comigration\_size' parameter) and perform an enrichment test (P-value of Fisher's exact test, 'pval\_threshold') to determine significance of enriched comigrations. The Fisher's exact test can only be done for two conditions at a time. The following 2x2 table was used to calculate the P-value from the Fisher's exact test. To evaluate significance of co-migrated proteins from cluster #1 in control to cluster #2 in test group,

	-	cluster #1 in control	other clusters in control
cluster #2 in test		65 (TP)	175 (FP)
other clusters in test		35 (FN)	979 (TN)

**Usage**

```
alluvial_enriched(clustering_result, selected_conditions,
  comigration_size = 0, pval_threshold = 1, pval_method = "fdr",
  cex = 0.7, alpha = 0.3)
```

**Arguments**

**clustering\_result**  
A list containing XINA clustering results. See [xina\\_clustering](#)

**selected\_conditions**  
A vector of condition names used in XINA clustering results. The number of selected conditions should be at least two.

**comigration\_size**  
The number of proteins comigrated together in the selected conditions of XINA clustering results. Default is 0

**pval\_threshold** This option is available only when you selected two conditions for comigration search.

**pval\_method** Method for p-value adjustment. See [p.adjust](#)

**cex** Scaling of fonts of category labels. Default if 0.7. See [alluvial](#)

**alpha** Transparency of the stripes. Default if 0.3. See [alluvial](#)

**Value**

A data frame containing comigrations and an alluvial plot showing comigrations

**Examples**

```
# load XINA example data
data(xina_example)

# Get the experimental conditions in the example data
classes <- as.vector(example_clusters$condition)

# Get comigrations without any thresholds
all_comigrations <- alluvial_enriched(example_clusters, classes)

# Get comigrations that have >= 5 size (the number of comigrated proteins)
all_cor_enriched <- alluvial_enriched(example_clusters, classes, comigration_size=5)

# Get all the comigrations between Control and Stimulus1
comigrations_Control_Stimulus1 <- alluvial_enriched(example_clusters,
c(classes[1],classes[2]))

# Get comigrations between Control and Stimulus1, that have >=5 size
comigrations_Control_Stimulus1_over5 <- alluvial_enriched(example_clusters,
c(classes[1],classes[2]), comigration_size=5)

# Get comigrations between Control and Stimulus1,
# that have >= 5 size and enrichment FDR <= 0.01
comigrations_Control_Stimulus1_pval0.01_size5 <- alluvial_enriched(example_clusters,
c(classes[1],classes[2]), comigration_size=5, pval_threshold=0.01)
```

```
# Get comigrations between Control and Stimulus1,
# that have >= 5 size and enrichment Benjamini & Yekutieli <= 0.01
comigrations_Control_Stimulus1_BY0.01_size5 <- alluvial_enriched(example_clusters,
c(classes[1],classes[2]), comigration_size=5, pval_threshold=0.01, pval_method="BY")
```

---

```
alluvial_enrichment_tests
      alluvial_enrichment_tests
```

---

### Description

Fisher's exact test to calculate the significance over all comigrations. The following 2x2 table was used to calculate p-value from Fisher's exact test. To evaluate significance of comigrated proteins from cluster #1 in control to cluster #2 in test condition,

	<b>cluster #1 in control</b>	<b>other clusters in control</b>
<b>cluster #2 in test</b>	65 (TP)	175 (FP)
<b>other clusters in test</b>	35 (FN)	979 (TN)

'alluvial\_enrichment\_tests' also provides another statistical methods including Hypergeometric test and Chi-square test.

### Usage

```
alluvial_enrichment_tests(count_table, c1, c2, non_cluster = 0,
  test_type = "fisher")
```

### Arguments

count_table	A data frame generated by using <a href="#">count</a> .
c1	A selected cluster in the first condition.
c2	A selected cluster in the second condition.
non_cluster	The cluster number for proteins that were not detected in a specific sample. Default is 0.
test_type	Enrichment test type. 'fisher' = Fisher's exact test, 'hyper' = Hypergeometric test, 'chisq' = Chi-square test

### Value

P-value of comigration enrichment test and 2x2 table information

---

calculate centrality scores  
*calculate centrality scores*

---

**Description**

'calculate centrality scores' computes network centrality scores

**Usage**

```
calculate_centrality_scores(net, centrality_type = "Degree")
```

**Arguments**

net                    protein-protein interaction network of igraph  
centrality\_type        the maximum number of clusters

**Value**

A vector of network centrality scores

---

default\_size            *default\_size*

---

**Description**

Calculate image size based on the number of clusters

**Usage**

```
default_size(max_cluster)
```

**Arguments**

max\_cluster            the maximum number of clusters

**Value**

A vector of plot width and height

---

draw\_alluvial\_plot     *draw\_alluvial\_plot*

---

## Description

'draw\_alluvial\_plot' draw a alluvial plot

## Usage

```
draw_alluvial_plot(clustering_result, selected_conditions, count_table,  
  alluvia_colors = NULL, cex = 0.7, alpha = 0.3)
```

## Arguments

**clustering\_result**     A list containing XINA clustering results. See [xina\\_clustering](#).

**selected\_conditions**     A vector of condition names used in XINA clustering results. The number of selected conditions should be at least two.

**count\_table**     A data frame generated by using [count](#).

**alluvia\_colors**     A vector containing the user-defined colors for each alluvium.

**cex**     Size of cluster number on block axis. Default if 0.7. See [alluvial](#).

**alpha**     Transparency of alluvia colors. Default is 0.3. See [alluvial](#).

## Value

An alluvial plot displaying comigrations and the data frame containing the input count\_table with colors.

## Examples

```
# load XINA example data  
data(xina_example)  
  
# get a vector of experimental conditions analyzed in the clustering results  
classes <- as.vector(example_clusters$condition)  
  
comigrations_size_over5 <- alluvial_enriched(example_clusters, classes, comigration_size=5)  
draw_alluvial_plot(example_clusters, classes, comigrations_size_over5)
```

---

example_clusters	<i>Randomly generated example datasets for XINA users. A dataset containing the XINA clustering results.</i>
------------------	--

---

### Description

- aligned. XINA clustering results aligned by conditions
- data\_column. Column names for data matrix
- out\_dir. Not available in this example dataset
- nClusters. The number of user-desired clusters. It's 30 in the example.
- max\_cluster. The number of clusters found in the dataset. It's 21 in the example.
- chosen\_model. The chosen covariance model for the example dataset. It's VEI in the example
- optimal\_BIC. BIC at the optimized clustering. It's 29473.57 in the example
- condition. The experimental conditions in the dataset.
- color\_for\_condition. The default color for the conditions that will be used in XINA plot drawing.
- color\_for\_clusters. The default color for the clusters that will be used in XINA clustering plot.
- norm\_method. The used normalization method to standardize the input data. It's "sum\_normalization" in the example.

### Format

A list with the example XINA clustering result

---

extract_data_column	<i>extract_data_column</i>
---------------------	----------------------------

---

### Description

Extract data column names from XINA clustering result

### Usage

```
extract_data_column(col_head_of_clustering)
```

### Arguments

col\_head\_of\_clustering  
Column names of XINA clustering result

### Value

A vector containing column names of data matrix



---

find\_similar\_clusters *find\_similar\_clusters*

---

**Description**

Compare clusters and find similar ones

**Usage**

```
find_similar_clusters(clustering_result, threshold = 0.95)
```

**Arguments**

clustering\_result

A list containing XINA clustering results. See [xina\\_clustering](#)

threshold

Pearson's r threshold to find similar ones

**Value**

Write a csv file containing similar clustering information based on the given Pearson's R threshold

---

generate\_count\_table *generate\_count\_table*

---

**Description**

Count the number of comigrated proteins using [count](#)

**Usage**

```
generate_count_table(clustering_result, selected_conditions,  
  comigration_size)
```

**Arguments**

clustering\_result

A list containing XINA clustering results. See [xina\\_clustering](#)

selected\_conditions

A vector of condition names used in XINA clustering results.

comigration\_size

The number of proteins comigrated together in the selected conditions of XINA clustering results. Default is 0.

**Value**

A data frame containing comigrations.

---

generate_superset	<i>generate_superset</i>
-------------------	--------------------------

---

**Description**

Merge input kinetics files

**Usage**

```
generate_superset(f_names, data_column, delim = ",",  
                 norm = "sum_normalization")
```

**Arguments**

f_names	A vector of .csv file paths containing kinetics data
data_column	A vector of column names containing data matrix
delim	The delimiter of input file (default is ',')
norm	The normalization method. It should be one of c('sum_normalization', 'zs-core'). Default is 'sum_normalization'.

**Value**

A data frame containing kinetics data obtained from files in the f\_names vector

---

get_colors	<i>get_colors</i>
------------	-------------------

---

**Description**

Generate color series for XINA graphics

**Usage**

```
get_colors(nClusters, set = "", colorset = NULL)
```

**Arguments**

nClusters	The number of clusters
set	Pre-defined color series set
colorset	manually defined color codes

**Value**

A vector for color code of XINA graphics

---

```
get_color_for_nodes  get_color_for_nodes
```

---

**Description**

Pre-defined 30 colors

**Usage**

```
get_color_for_nodes()
```

**Value**

A vector for color code of XINA graphics

---

```
get_comigrations_by_name
      get_comigrations_by_name
```

---

**Description**

'get\_comigrations\_by\_name' finds proteins comigrated with the given proteins

**Usage**

```
get_comigrations_by_name(clustering_result, selected_conditions,
  protein_list, cex = 0.7, alpha = 0.3)
```

**Arguments**

**clustering\_result** A list containing XINA clustering results. See [xina\\_clustering](#)

**selected\_conditions** A vector of condition names used in XINA clustering results. The number of selected conditions should be at least two.

**protein\_list** A vector containing gene names.

**cex** Size of cluster number on block axis. Default if 0.7. See [alluvial](#)

**alpha** Transparency of alluvia colors. Default is 0.3. See [alluvial](#)

**Value**

An alluvial plot displaying comigrations and the data frame containing comigrations of the input proteins

**Examples**

```
# load XINA example data
data(xina_example)

# the clustering result table
all_proteins <- as.character(example_clusters$aligned$`Gene name`)
# get a vector of experimental conditions analyzed in the clustering results
classes <- as.vector(example_clusters$condition)

comigrated_prots_all <- get_comigrations_by_name(example_clusters, classes, all_proteins[1:3])
```

---

```
get_condition_biased_comigrations
      get_condition_biased_comigrations
```

---

**Description**

get comigrations that at least one biased cluster is involved in. Biased clusters are defined by

**Usage**

```
get_condition_biased_comigrations(clustering_result, count_table = NULL,
  selected_conditions, condition_composition, threshold_percent = 50,
  color_for_null = "gray", color_for_highly_matched = "red4",
  cex = 0.7, alpha = 0.3)
```

**Arguments**

**clustering\_result**  
A list containing XINA clustering results. See [xina\\_clustering](#)

**count\_table**  
A data frame generated by using [count](#). If count\_table is NULL (by default), XINA will consider all the comigrations.

**selected\_conditions**  
A vector of condition names used in XINA clustering results. The number of selected conditions should be at least two.

**condition\_composition**  
The resulting data frame of 'plot\_condition\_compositions'. See [plot\\_condition\\_compositions](#).

**threshold\_percent**  
Default is 50. The percentage threshold for finding condition-biased clusters

**color\_for\_null**  
A color for non-condition-biased comigrations. Default is 'gray'

**color\_for\_highly\_matched**  
A color for comigrations that are involved with more than two condition-biased clusters. Default is 'red4'

**cex**  
Size of cluster number on block axis. Default if 0.7. See [alluvial](#).

**alpha**  
Transparency of alluvia colors. Default is 0.3. See [alluvial](#).

**Value**

An alluvial plot displaying comigrations and the data frame containing condition-biased comigrations.

**Examples**

```
# load XINA example data
data(xina_example)

# get a vector of experimental conditions analyzed in the clustering results
conditions <- as.vector(example_clusters$condition)

# get condition composition information
condition_composition <- plot_condition_compositions(example_clusters)

comigrations_size10 <- alluvial_enriched(example_clusters, conditions, comigration_size=10)
# Finding condition-biased comigrations by 50% threshold
condition_biased_comigrations <-
get_condition_biased_comigrations(clustering_result=example_clusters,
count_table=comigrations_size10, selected_conditions=conditions,
condition_composition=condition_composition)

# Finding condition-biased comigrations by 70% threshold
condition_biased_comigrations <-
get_condition_biased_comigrations(clustering_result=example_clusters,
count_table=comigrations_size10, selected_conditions=conditions,
condition_composition=condition_composition,
threshold_percent=70)
```

---

get\_layout

*get\_layout*

---

**Description**

Get igraph layout by the number of nodes

**Usage**

```
get_layout(subnet_condition)
```

**Arguments**

```
subnet_condition
  A igraph sub-network
```

**Value**

igraph network layout

---

`get_mTOR_proteins`      *get\_mTOR\_proteins*

---

**Description**

Get mTOR pathway genes

**Usage**

```
get_mTOR_proteins(time_points, conditions)
```

**Arguments**

`time_points`      A vector containing time points of the data matrix  
`conditions`      A vector containing condition information, for example normal, disease and drug treated disease.

**Value**

A vector containing mTOR pathway gene names

---

`get_random_data`      *get\_random\_data*

---

**Description**

Get randomized time-series data

**Usage**

```
get_random_data(time_points, conditions, num_total, percent.sign = 0.1,
  equal = TRUE)
```

**Arguments**

`time_points`      A vector containing time points of the data matrix  
`conditions`      A vector containing condition information, for example normal, disease and drug treated disease.  
`num_total`      The number of total proteins to be generated  
`percent.sign`      Percentage of differentially expressed proteins. Ignored when `equal=FALSE`.  
`equal`      If `equal` is `TRUE`, all the conditions will have numbers between 0 and 1. If it is `FALSE`, the first three conditions will have different ranges. First condition will have numbers from 0.3 to 0.4. Second condition will have numbers from 0.6 to 0.8. Third condition will have numbers from 0.3 to 0.5. Other conditions will have numbers from 0 to 1.

**Value**

A list containing randomly generated data matrix

---

get_stats	<i>get_stats</i>
-----------	------------------

---

**Description**

Calculate statistics of the given data for XINA network analysis

**Usage**

```
get_stats(centrality_results, na.rm = FALSE)
```

**Arguments**

`centrality_results` Network centrality score data frame calculated by XINA network module

`na.rm` If it is FALSE, no exclusion of NA values.

**Value**

A data frame containing statistics of XINA network centrality scores

---

get_theme_blank	<i>get_theme_blank</i>
-----------------	------------------------

---

**Description**

Predefined ggplot theme for removing ticks, titles and labels of X and Y axis

**Usage**

```
get_theme_blank()
```

**Value**

A ggplot theme





---

gn_desc	<i>A character vector containing 19,396 human gene descriptions This is for the random data generation of XINA</i>
---------	--

---

**Description**

- Human gene description corresponding to 'gn' vector

**Format**

A character vector containing 19,396 human gene descriptions

**Source**

<https://www.ncbi.nlm.nih.gov/gene>

---

hprd_ppi	<i>Protein-protein interaction resource downloaded from HPRD DB A data frame containing HRPD protein-protein interaction data</i>
----------	---

---

**Description**

- gene\_symbol\_1. Gene name interacting with gene name in 'gene\_symbol\_2'
- gene\_symbol\_2. Gene name interacting with gene name in 'gene\_symbol\_1'
- Experiment\_type. Experimental or computational methods supporting the interaction

**Format**

A data frame containing HRPD protein-protein interaction data

**Source**

<http://www.hprd.org/>

---

length2	<i>length2</i>
---------	----------------

---

**Description**

Customized function for vector length calculation

**Usage**

```
length2(x, na.rm = FALSE)
```

**Arguments**

x	A vector
na.rm	If it is FALSE, no exclusion of NA values.

**Value**

A vector length

---

load\_previous\_results *load\_previous\_results*

---

**Description**

Get previous XINA clustering results to R space

**Usage**

```
load_previous_results(clustering_dir = getwd(), data_column = NULL,  
  fp_clusters = "xina_clusters.csv")
```

**Arguments**

clustering_dir	The directory path of XINA clustering results
data_column	A vector containing column names of data matrix
fp_clusters	File path of XINA clustering results

**Value**

Comma-separated file containing aligned XINA clustering results.

**Examples**

```
# Load XINA's example data  
data(xina_example)  
write.csv(example_clusters$aligned, "xina_clusters_aligned.csv")  
write.csv(example_clusters$clusters, "xina_clusters.csv")  
  
# Reload the clustering result  
example_clusters_reloaded <- load_previous_results(".")
```

---

make\_random\_xina\_data *make\_random\_xina\_data*

---

## Description

Generate random proteomics dataset for testing XINA 'make\_random\_xina\_data' will make random proteomics data for XINA test. The generated data will have three conditions and seven time points, c("0hr", "2hr", "6hr", "12hr", "24hr", "48hr", "72hr").

## Usage

```
make_random_xina_data(n = 500, mtor = TRUE, time_points = c("0hr",  
  "2hr", "6hr", "12hr", "24hr", "48hr", "72hr"),  
  conditions = c("Control", "Stimulus1", "Stimulus2"))
```

## Arguments

n	The number of proteins for one condition. Default is 500.
mtor	If it is TRUE (default), mTOR pathway genes will be significant. If it is FALSE, randomly selected genes will be significant in first three conditions.
time_points	A vector containing time points of the data matrix
conditions	A vector containing condition information, for example normal, disease and drug treated disease.

## Value

Three comma-separated files containing time-series data for XINA

## Examples

```
make_random_xina_data()  
g1 <- read.csv("Control.csv", check.names=FALSE,  
  stringsAsFactors = FALSE)  
g2 <- read.csv("Stimulus1.csv", check.names=FALSE,  
  stringsAsFactors = FALSE)  
g3 <- read.csv("Stimulus2.csv", check.names=FALSE,  
  stringsAsFactors = FALSE)  
  
head(g1)  
head(g2)  
head(g3)
```

---

mutate_colors	<i>mutate_colors</i>
---------------	----------------------

---

## Description

'mutate\_colors' generates new color scheme for XINA clustering plot based on condition composition results ([plot\\_condition\\_compositions](#)). If any clusters have higher percentage than the 'threshold\_percent', XINA will assign new colors in accordance to 'color\_for\_condition'. If not, XINA will give 'gray' color or user-defined color via 'null\_color' parameter.

## Usage

```
mutate_colors(condition_composition, color_for_condition,  
              null_color = "gray", threshold_percent = 50)
```

## Arguments

`condition_composition`  
A data frame generated by [plot\\_condition\\_compositions](#)

`color_for_condition`  
A vector like 'color\_for\_condition' of [xina\\_clustering](#)

`null_color`      Default is 'gray'. This color is for clusters that are not biased to any of experimental conditions

`threshold_percent`  
Default is 50. The percentage threshold for giving new colors

## Value

A data frame containing statistics of XINA network centrality scores

## Examples

```
# load XINA example data  
data(xina_example)  
  
# Plot condition composition pie-chart with default option  
condition_composition <- plot_condition_compositions(example_clusters)  
example_clusters$color_for_clusters <- mutate_colors(condition_composition,  
example_clusters$color_for_condition)  
plot_clusters(example_clusters, xval=c(0,2,6,12,24,48,72), xylab=FALSE)
```

---

organize_clusters	<i>organize_clusters</i>
-------------------	--------------------------

---

**Description**

Organize XINA clustering information by gene name

**Usage**

```
organize_clusters(clustering_dir = getwd(), super_ds, file_out = TRUE)
```

**Arguments**

clustering_dir	The directory path of XINA clustering results
super_ds	XINA clusters
file_out	If it is TRUE, it writes the aligned clustering informaion to "xina_clusters_aligned.csv" file.

**Value**

Comma-separated file containing aligned XINA clustering results.

---

plot_clusters	<i>plot_clusters</i>
---------------	----------------------

---

**Description**

Draw all the clustering results. 'plot\_clusters' draws two plots, scaled and unscaled line graphs. Scaled graphs have same y limits that are 0 to 1 by default, but can be changed via 'y\_lim' parameter.

**Usage**

```
plot_clusters(clustering_result, y_lim = NULL, xval = NULL,
             xylab = TRUE, ggplot_theme = NULL)
```

**Arguments**

clustering_result	A list containing XINA clustering results. See <a href="#">xina_clustering</a>
y_lim	Y axis limit. If you set y_lim=c(0,1), 'plot_clusters' will plot line graphs scaled from 0 to 1 in y-axis Default is NULL, which means unscaled line graphs.
xval	Change X axis values and labels. Default is data_column of the clustering result list
xylab	If it is FALSE, x and y labels will be blank. If it is TRUE (default), x and y labels will be shown.
ggplot_theme	This is ggplot theme to modify XINA clustering plot.

**Value**

Line graphs of all the clusters

**Examples**

```
library(ggplot2)

# load XINA example data
data(xina_example)

# Draw clustering plots
plot_clusters(example_clusters)

# Apply theme to the clustering plot
theme1 <- theme(title=element_text(size=8, face='bold'),
axis.text.x = element_text(size=7),
axis.text.y = element_blank(),
axis.ticks.x = element_blank(),
axis.ticks.y = element_blank(),
axis.title.x = element_blank(),
axis.title.y = element_blank())
plot_clusters(example_clusters, ggplot_theme=theme1)
```

---

plot\_clusters\_all      *plot\_clusters\_all*

---

**Description**

Draw line graphs of all the proteins in the given dataset

**Usage**

```
plot_clusters_all(clustering_result, selected_condition = NULL)
```

**Arguments**

clustering\_result

A list containing XINA clustering results. See [xina\\_clustering](#)

selected\_condition

A condition name to draw the kinetics plot

**Value**

a list containing clustering results and pdf file containing a BIC plot in current working directory.

**Examples**

```
# load XINA example data
data(xina_example)

# Plot kinetics of all the proteins in Control
```

```
plot_clusters_all(example_clusters, selected_condition="Control")

# Plot kinetics of all the proteins in Stimulus1
plot_clusters_all(example_clusters, selected_condition="Stimulus1")

# Plot kinetics of all the proteins in Stimulus2
plot_clusters_all(example_clusters, selected_condition="Stimulus2")

# Plot kinetics of all the proteins in three data
plot_clusters_all(example_clusters)
```

---

```
plot_condition_compositions
      plot_condition_compositions
```

---

### Description

computes condition composition of the XINA clustering results and draws pie-charts.

### Usage

```
plot_condition_compositions(clustering_result, bullseye = FALSE,
                             ggplot_theme = NULL)
```

### Arguments

clustering_result	A list containing XINA clustering results. See <a href="#">xina_clustering</a>
bullseye	If it is TRUE, draw bullseye plot instead of the pie-chart. Default is FALSE
ggplot_theme	This is ggplot theme to modify condition composition pie-chart and bulles eye plots.

### Value

A condition composition plot and a data frame containing condition compositions of the clusters

### Examples

```
# load XINA example data
data(xina_example)

# Plot condition composition pie-chart with default option
plot_condition_compositions(example_clusters)

# Make a new color code for conditions
condition_colors <- c("tomato", "steelblue1", "gold")
names(condition_colors) <- example_clusters$condition
example_clusters$color_for_condition <- condition_colors

# Draw condition composition pie-chart with the new color code
plot_condition_compositions(example_clusters)
```

```
# Draw condition composition bullseye plot
plot_condition_compositions(example_clusters, bullseye = TRUE)
```

---

```
plot_enrichment_results
      plot_enrichment_results
```

---

## Description

Plot GO and KEGG enrichment results

## Usage

```
plot_enrichment_results(enriched_results,
  term_description = "term_description", sig_score = "pvalue",
  num_terms = 0, get_log = TRUE)
```

## Arguments

enriched_results	GO or KEGG enrichment results. See <a href="#">xina_enrichment</a> and <a href="#">xina_enrichment</a>
term_description	Description of terms to be drawn on Y axis. Default is "term_description" of XINA enrichment results.
sig_score	significant score to plot on X axis. Default is "pvalue".
num_terms	The number of terms to be plotted. Default is 0, which means no limit.
get_log	If this is TRUE, 'plot_enrichment_results' will take -log10 of p-values.

## Value

ggplot bar graph

## Examples

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

# load XINA example data
data(xina_example)

# Get STRING database for protein-protein intereaction information
string_db <- STRINGdb$new( version="10", species=9606,
  score_threshold=0, input_directory="" )
string_db

# XINA analysis with STRING DB
xina_result <- xina_analysis(example_clusters, string_db)

# Select proteins that showed cluster #1 in the Stimulus2 condition
subgroup <- subset(example_clusters$aligned, Stimulus2==1)
```



```

protein_list <- as.vector(subgroup$`Gene name`)

# Enrichment test and get significantly enriched functional terms
# that have adjusted p-value less than 0.1
kegg_enriched <- xina_enrichment(string_db, protein_list,
enrichment_type = "KEGG", pval_threshold=0.1)
plot_enrichment_results(kegg_enriched$KEGG, num_terms=10)

## End(Not run)

```

---

plot\_NA

*plot\_NA*


---

### Description

Draw NULL plot

### Usage

```
plot_NA()
```

### Value

a empty plot

---

rank\_centrality

*rank\_centrality*


---

### Description

Give ranks based on network centrality scores

### Usage

```
rank_centrality(centrality_score, type, num_breaks = 5)
```

### Arguments

centrality\_score      Network centrality score matrix  
type                    Network centrality score type, such as 'Eigenvector'  
num\_breaks            The number of ranks

### Value

A vector containing ranks

---

string_example	<i>Protein-protein interaction resource downloaded from STRING DB for XINA's example dataset A data frame containing protein-protein interactions</i>
----------------	---

---

### Description

- gene\_symbol\_1. Gene name interacting with gene name in 'gene\_symbol\_2'
- gene\_symbol\_2. Gene name interacting with gene name in 'gene\_symbol\_1'
- PPI\_Source. Data original source

### Format

A data frame containing STRING protein-protein interaction data

### Source

<https://string-db.org/>

---

xina_analysis	<i>xina_analysis</i>
---------------	----------------------

---

### Description

xina\_analysis is to analyze protein-protein interaction(PPI) networks using STRINGdb and igraph R package. This module computes PPI networks within each XINA clusters.

### Usage

```
xina_analysis(clustering_result, ppi_db, is_stringdb = TRUE,
              flag_simplify = TRUE, node_shape = "sphere",
              num_clusters_in_row = 5, img_size = NULL, img_qual = 300)
```

### Arguments

clustering_result	A list containing XINA clustering results. See <a href="#">xina_clustering</a>
ppi_db	STRINGdb object
is_stringdb	If it is TRUE (default), XINA will process 'ppi_db' as STRINGdb, but if it is FALSE, XINA will accept your 'ppi_db' as it is. You can make your own igraph network using customized PPI information instead of STRINGdb.
flag_simplify	If it is TRUE (default), XINA will exclude unconnected proteins
node_shape	You can choose node shape. Default is "sphere". See <a href="#">shapes</a>
num_clusters_in_row	The number of clusters in a row on the XINA network plot. Default is 5.
img_size	Set the image size. For width=1000 and height=1500, it is img_size=c(1000,1500).
img_qual	Set the image resolution. Default is 300.

**Value**

A PNG file (XINA\_Cluster\_Networks.png) displaying PPI network plots of all the clusters and a list containing XINA network analysis results.

Item	Description
All_network	PPI network of all the input proteins
Sub_network	A list containing PPI networks of each clusters
Data	XINA clustering results. See <a href="#">xina_clustering</a>
Nodes	A list of proteins in each cluster
Conditions	A list of experimental condition of proteins in each cluster
Titles	A list of plot titles for XINA plotting
out_dir	A directory path storing XINA network analysis results
is_stringdb	False = different PPI DB and TRUE = STRING DB

**Examples**

```
## Not run:
# load XINA example data
data(xina_example)

# use the following code for utilizing up-to-date STRING DB
tax_id <- 9606 # for human
# tax_id <- 10090 # for mouse
library(STRINGdb)
library(igraph)
string_db <- STRINGdb$new( version='10', species=tax_id, score_threshold=0, input_directory='' )
string_db
xina_result <- xina_analysis(example_clusters, string_db, flag_simplify=FALSE)

# Run XINA with a protein-protein interaction edgelist
data(HPRD)
net_all <- simplify(graph_from_data_frame(d=hprd_ppi, directed=FALSE),
remove_multiple = FALSE, remove_loops = TRUE)
xina_result <- xina_analysis(example_clusters, net_all, is_stringdb=FALSE, flag_simplify=FALSE)

## End(Not run)
```

---

xina\_clustering      *xina\_clustering*

---

**Description**

Clustering multiplexed time-series omics data to find co-abundance profiles

**Usage**

```
xina_clustering(f_names, data_column, out_dir = getwd(),
nClusters = 20, norm = "sum_normalization", chosen_model = "")
```

**Arguments**

f_names	A vector containing input file (.csv) paths
data_column	A vector containing column names (1st row of the input file) of data matrix
out_dir	A directory path for saving clustering results. (default: out_dir=getwd())
nClusters	The number of desired maximum clusters
norm	Default is "sum_normalization". Sum-normalization is to divide the data matrix by row sum. If you want to know more about sum-normalization, see <a href="https://www.ncbi.nlm.nih.gov/pubmed/19861354">https://www.ncbi.nlm.nih.gov/pubmed/19861354</a> . "zscore" is to calculate Z score for each protein. See <a href="#">scale</a> .
chosen_model	You can choose a specific model rather than testing all the models that are available in mclust. <a href="#">mclustModelNames</a> If you want k-means clustering instead of the model-based clustering, use "kmeans" here.

**Value**

a plot containing a BIC plot in current working directory and a list containing below information:

Item	Description
clusters	XINA clustering results
aligned	XINA clustering results aligned by ID
data_column	Data matrix column names
out_dir	The directory path containing XINA results
nClusters	The number of clusters desired by user
max_cluster	The number of clusters optimized by BIC
chosen_model	The used covariance model for model-based clustering
optimal_BIC	BIC of the optimized covariance model
condition	Experimental conditions of the user input data
color_for_condition	Colors assigned to each experimental conditions which is used for condition composition plot
color_for_clusters	Colors assigned to each clusters which is used for XINA clustering plot
norm_method	Used normalization method

**Examples**

```
# Generate random multiplexed time-series data
random_data_info <- make_random_xina_data()

# Data files
data_files <- paste(random_data_info$conditions, ".csv", sep='')

# time points of the data matrix
data_column <- random_data_info$time_points

# mclust requires the fixed random seed to get reproduce the clustering results
set.seed(0)

# Run the model-based clustering to find co-abundance profiles
example_clusters <- xina_clustering(data_files, data_column=data_column,
nClusters=30)

# Run k-means clustering to find co-abundance profiles
example_clusters <- xina_clustering(data_files, data_column=data_column,
```

```
nClusters=30,
chosen_model="kmeans")
```

---

xina_enrichment	<i>xina_enrichment</i>
-----------------	------------------------

---

## Description

xina\_enrichment conducts functional enrichment tests using gene ontology or KEGG pathway terms for a given protein list

## Usage

```
xina_enrichment(string_db, protein_list, enrichment_type = "GO",
  pval_threshold = 0.05, methodMT = "fdr")
```

## Arguments

string_db	STRINGdb object
protein_list	A vector of gene names to draw protein-protein interaction network.
enrichment_type	A functional annotation for the enrichment test. 'enrichment_type' should be one of 'GO' and 'KEGG',
pval_threshold	P-value threshold to get significantly enriched terms from the given proteins
methodMT	Method for p-value adjustment. See <a href="#">get_enrichment</a> . Default is 'fdr'.

## Value

A list of data frames containing enrichment results

## Examples

```
## Not run:
library(STRINGdb)
library(Biobase)

# load XINA example data
data(xina_example)

# Get STRING database for protein-protein intereaction information
string_db <- STRINGdb$new( version="10", species=9606, score_threshold=0, input_directory="" )
string_db

# XINA analysis with STRING DB
xina_result <- xina_analysis(example_clusters, string_db)

# Select proteins that showed cluster #1 in the Stimulus2 condition
subgroup <- subset(example_clusters$aligned, Stimulus2==1)
protein_list <- as.vector(subgroup$`Gene name`)

# Enrichment test using KEGG pathway terms that have adjuseted p-value less than 0.1
```

```

kegg_enriched <- xina_enrichment(string_db, protein_list,
enrichment_type = "KEGG", pval_threshold=0.1)
plot_enrichment_results(kegg_enriched$KEGG, num_terms=10)

# Enrichment test using GO terms that have adjusted p-value less than 0.1
go_enriched <- xina_enrichment(string_db, protein_list,
enrichment_type = "GO", pval_threshold=0.1)
plot_enrichment_results(go_enriched$Component, num_terms=10)

## End(Not run)

```

---

xina\_plot\_all

*xina\_plot\_all*


---

## Description

xina\_plot\_all is to draw protein-protein interaction network plots of all the clusters

## Usage

```

xina_plot_all(xina_result, clustering_result, condition = "all",
centrality_type = NULL, flag_simplify = TRUE, num_breaks = 5,
layout_specified = "", vertex_label_flag = FALSE,
vertex.label.color = "black", vertex.color = "", edge.color = NULL,
vertex.label.dist = 0.6, vertex.label.cex = 0.8,
edge.arrow.size = 0.4, vertex.size = 10, vertex.shape = "sphere",
legend_location = "bottom", num_clusters_in_row = 5,
flag_unknown_only = FALSE, img_size = NULL, img_qual = 300)

```

## Arguments

**xina\_result** A list containing XINA network analysis results. See [xina\\_analysis](#)

**clustering\_result** A list containing XINA clustering results. See [xina\\_clustering](#)

**condition** Default is 'all', which means use all the proteins to draw graphs. If you specify the experimental condition name used for XINA clustering, xina\_plot\_all will draw graphs using specific condition's proteins.

**centrality\_type** 'centrality\_type' should be one of c('Degree', 'Eigenvector', 'Hub', 'Authority', 'Closeness', 'Betweenness')

Centrality score	igraph function
Degree	<a href="#">degree</a>
Eigenvector	<a href="#">eigen_centrality</a>
Hub	<a href="#">hub_score</a>
Authority	<a href="#">authority_score</a>
Closeness	<a href="#">closeness</a>
Betweenness	<a href="#">betweenness</a>

`flag_simplify` If it is TRUE (default), XINA will exclude unconnected proteins

`num_breaks` 'num\_breaks' is the number of ranks based on network centrality. Default is 5.

`layout_specified`  
This can change network layout. 'layout\_specified' should be one of c('sphere', 'star', 'gem', 'tree', 'circle', 'random', 'nicely'). XINA's layouts are based on igraph's layout. See [layout\\_](#)

Layout	igraph layout name
sphere	<a href="#">layout_on_sphere</a>
star	<a href="#">layout_as_star</a>
gem	<a href="#">layout_with_gem</a>
tree	<a href="#">layout_as_tree</a>
circle	<a href="#">layout_in_circle</a>
random	<a href="#">layout_randomly</a>
nicely	<a href="#">layout_nicely</a>

Default is 'layout\_nicely' of igraph

`vertex_label_flag`  
If `vertex_label_flag` is TRUE (default), igraph network graphs will be labeled by gene names. If `vertex_label_flag` is FALSE, igraph network graphs will be drawn without labels.

`vertex.label.color`  
Color of labels. Default is black.

`vertex.color` Color of nodes. Default is pink.

`edge.color` Color of edges. Default is pink.

`vertex.label.dist`  
Distance between node and label. Default is 0.6.

`vertex.label.cex`  
Size of labels. Default is 0.8.

`edge.arrow.size`  
Size of edges. Default is 0.4.

`vertex.size` Size of nodes. Default is 10.

`vertex.shape` You can choose node shape. Default is 'sphere'. See [shapes](#)

`legend_location`  
If `centrality_type` is chosen, `xina_plot_single` add the color legend guiding rank of nodes based on the centrality score. Default is 'bottomright', but you can choose one of these 'bottomright', 'bottom', 'bottomleft', 'left', 'topleft', 'top', 'topright', 'right' and 'center'.

`num_clusters_in_row`  
The number of clusters in a row on the XINA network plot. Default is 5.

`flag_unknown_only`  
If this is TRUE, 'xina\_plot\_all' will plot proteins that do not have any protein-protein interaction in the given database.

`img_size` Set the image size. For width=1000 and height=1500, it is `img_size=c(1000,1500)`. Default is `c(3000,3000)`.

`img_qual` Set the image resolution. Default is 300.

**Value**

PNG images of PPI network plots of all the clusters

**Examples**

```
## the following code is to show how it works quickly
## load XINA example data
data(xina_example)

## load the previously processed XINA analysis results
# if you want to learn how to run 'xina_analysis', please see \link[XINA]{xina_analysis}
data(xina_result_example)

# XINA network plots
xina_plot_all(xina_result_example, example_clusters)

# XINA network plots for Control condition
xina_plot_all(xina_result_example, example_clusters, condition='Control')
```

---

xina\_plot\_bycluster    *xina\_plot\_bycluster*

---

**Description**

xina\_plot\_bycluster is to draw protein-protein interaction network plots of each cluster

**Usage**

```
xina_plot_bycluster(xina_result, clustering_result, cl = NULL,
  condition = "all", flag_legend = TRUE, centrality_type = NULL,
  flag_simplify = TRUE, layout_specified = "",
  vertex_label_flag = TRUE, vertex.label.dist = 0.6,
  vertex.label.cex = 0.8, edge.arrow.size = 0.4, vertex.size = 10,
  vertex.shape = "sphere", vertex.color = "",
  edge.color = "darkgray", legend_location = "bottom",
  flag_unknown_only = FALSE)
```

**Arguments**

xina_result	A list containing XINA network analysis results. See <a href="#">xina_analysis</a>
clustering_result	A list containing XINA clustering results. See <a href="#">xina_clustering</a>
cl	Cluster number in the XINA clustering results
condition	Default is 'all', which means use all the proteins to draw graphs. If you specify the experimental condition name used for XINA clustering,
flag_legend	If it is TRUE, a legend will be printed out together.
centrality_type	'centrality_type' should be one of c('Degree', 'Eigenvector', 'Hub', 'Authority', 'Closeness', 'Betweenness')



Centrality score	igraph function
Degree	<a href="#">degree</a>
Eigenvector	<a href="#">eigen_centrality</a>
Hub	<a href="#">hub_score</a>
Authority	<a href="#">authority_score</a>
Closeness	<a href="#">closeness</a>
Betweenness	<a href="#">betweenness</a>

`flag_simplify` If it is TRUE (default), XINA will exclude unconnected proteins

`layout_specified`

This can change network layout. 'layout\_specified' should be one of c('sphere', 'star', 'gem', 'tree', 'circle', 'random', 'nicely'). XINA's layouts are based on igraph's layout. See [layout\\_](#)

Layout	igraph layout name
sphere	<a href="#">layout_on_sphere</a>
star	<a href="#">layout_as_star</a>
gem	<a href="#">layout_with_gem</a>
tree	<a href="#">layout_as_tree</a>
circle	<a href="#">layout_in_circle</a>
random	<a href="#">layout_randomly</a>
nicely	<a href="#">layout_nicely</a>

Default is 'layout\_nicely' of igraph

`vertex_label_flag`

If `vertex_label_flag` is TRUE (default), igraph network graphs will be labeled by gene names. If `vertex_label_flag` is FALSE, igraph network graphs will be drawn without labels.

`vertex.label.dist`

Distance between node and label. Default is 0.6

`vertex.label.cex`

Size of labels. Default is 0.8

`edge.arrow.size`

Size of edges. Default is 0.4

`vertex.size` Size of nodes. Default is 10

`vertex.shape` You can choose node shape. Default is 'sphere'. See [shapes](#)

`vertex.color` Color of nodes. Default is pink.

`edge.color` Color of edges. Default is pink.

`legend_location`

If `centrality_type` is chosen, `xina_plot_single` add the color legend guiding rank of nodes based on the centrality score. Default is 'bottomright', but you can choose one of these 'bottomright', 'bottom', 'bottomleft', 'left', 'topleft', 'top', 'topright', 'right' and 'center'.

`flag_unknown_only`

If this is TRUE, 'xina\_plot\_bycluster' will plot proteins that do not have any protein-protein interaction in the given database

**Value**

A PNG file (XINA\_Cluster\_Networks.png) displaying protein-protein interaction network plots of all the clusters and a list containing XINA network analysis results

PNG images of PPI network plots of all the clusters

**Examples**

```
## the following code is to show how it works quickly
## load XINA example data
data(xina_example)

## load the previously processed XINA analysis results
# if you want to learn how to run 'xina_analysis', please see \link[XINA]{xina_analysis}
data(xina_result_example)

# plot cluster #1
xina_plot_bycluster(xina_result_example, example_clusters, cl=1)

# plot PPI network of Control condition in cluster #1
xina_plot_bycluster(xina_result_example, example_clusters, cl=1, condition='Control')
```

---

xina\_plot\_single

*xina\_plot\_single*


---

**Description**

xina\_plot\_single draws protein-protein interaction network plot for given 'protein\_list'.

**Usage**

```
xina_plot_single(xina_result, protein_list, centrality_type = NULL,
  layout_specified = "", vertex_label_flag = TRUE, main = NULL,
  vertex.label.color = "black", vertex.color = NA,
  edge.color = "darkgray", vertex.label.dist = 0.6,
  vertex.label.cex = 0.8, edge.arrow.size = 0.4, vertex.size = 10,
  vertex.shape = "sphere", legend_location = "bottom",
  num_breaks = 5, digits_round_up = 5, flag_simplify = TRUE,
  flag_legend = TRUE)
```

**Arguments**

xina\_result      A list containing XINA network analysis results. See [xina\\_analysis](#)

protein\_list     A vector of gene names to draw a protein-protein interaction network graph.

centrality\_type    'centrality\_type' should be one of c('Degree', 'Eigenvector', 'Hub', 'Authority', 'Closeness', 'Betweenness')

Centrality score	igraph function
Degree	<a href="#">degree</a>
Eigenvector	<a href="#">eigen_centrality</a>

Hub	<a href="#">hub_score</a>
Authority	<a href="#">authority_score</a>
Closeness	<a href="#">closeness</a>
Betweenness	<a href="#">betweenness</a>

layout\_specified

This can change network layout. 'layout\_specified' should be one of c('sphere', 'star', 'gem', 'tree', 'circle', 'random', 'nicely'). XINA's layouts are based on igraph's layout. See [layout\\_](#)

Layout	igraph layout name
sphere	<a href="#">layout_on_sphere</a>
star	<a href="#">layout_as_star</a>
gem	<a href="#">layout_with_gem</a>
tree	<a href="#">layout_as_tree</a>
circle	<a href="#">layout_in_circle</a>
random	<a href="#">layout_randomly</a>
nicely	<a href="#">layout_nicely</a>

Default is 'layout\_nicely' of igraph

vertex\_label\_flag

If vertex\_label\_flag is TRUE (default), igraph network graphs will be labeled by gene names. If vertex\_label\_flag is FALSE, igraph network graphs will be drawn without labels.

main

Title of network figure. IF it is NULL (default), it will be the number of plotted proteins.

vertex.label.color

Color of labels. Default is black.

vertex.color

Color of nodes. Default is pink.

edge.color

Color of edges. Default is pink.

vertex.label.dist

Distance between node and label. Default is 0.6.

vertex.label.cex

Size of labels. Default is 0.8.

edge.arrow.size

Size of edges. Default is 0.4.

vertex.size

Size of nodes. Default is 10.

vertex.shape

You can choose node shape. Default is 'sphere'. See [shapes](#).

legend\_location

If centrality\_type is chosen, 'xina\_plot\_single' adds the color legend guiding rank of nodes based on the centrality score. Default is 'bottomright', but you can choose one of these 'bottomright', 'bottom', 'bottomleft', 'left', 'topleft', 'top', 'topright', 'right' and 'center'.

num\_breaks

'num\_breaks' is the number of ranks based on network centrality. Default is 5.

digits\_round\_up

See [Round](#).

flag\_simplify

If it is TRUE (default), XINA will exclude unconnected proteins.

flag\_legend

If it is TRUE, a legend will be printed out together.

**Value**

A PNG file (XINA\_Cluster\_Networks.png) displaying protein-protein interaction network plots of all the clusters and a list containing XINA network analysis results

**Examples**

```
## the following code is to show how it works quickly
## load XINA example data
data(xina_example)

## load the previously processed XINA analysis results
# if you want to learn how to run 'xina_analysis', please see \link[XINA]{xina_analysis}
data(xina_result_example)

# get gene names that are clustered to #21 in "Stimulus2" condition
subgroup <- subset(example_clusters$aligned, Stimulus2==21)
protein_list <- subgroup$`Gene name`

# Calculate protein-protein interaction network
xina_plot_single(xina_result_example, protein_list)

# Calculate protein-protein interaction network and Eigenvector centrality
eigen_info <- xina_plot_single(xina_result_example, protein_list, centrality_type='Eigenvector')
```

---

xina_result_example	<i>Previously processed xina analysis using XINA's random example data A list containing 'xina_analysis' results</i>
---------------------	--

---

**Description**

- All\_network. PPI network of all the input proteins
- Sub\_network. A list containing PPI networks of each clusters
- Data. XINA clustering results. See [xina\\_clustering](#)
- Nodes. A list of proteins in each cluster
- Conditions. A list of experimental condition of proteins in each cluster
- Titles. A list of plot titles for XINA plotting
- out\_dir. A directory path storing XINA network analysis results
- is\_stringdb. False = different PPI DB and TRUE = STRING DB

**Format**

A data frame containing STRING protein-protein interaction data

**Source**

XINA

# Index

add\_legend, 3  
alluvial, 4, 7, 11, 12  
alluvial\_enriched, 3  
alluvial\_enrichment\_tests, 5  
authority\_score, 30, 33, 35  
  
betweenness, 30, 33, 35  
  
calculate\_centrality\_scores, 6  
closeness, 30, 33, 35  
count, 5, 7, 9, 12  
  
default\_size, 6  
degree, 30, 33, 34  
draw\_alluvial\_plot, 7  
  
eigen\_centrality, 30, 33, 34  
example\_clusters, 8  
extract\_data\_column, 8  
  
find\_similar\_clusters, 9  
  
generate\_count\_table, 9  
generate\_superset, 10  
get\_color\_for\_nodes, 11  
get\_colors, 10  
get\_comigrations\_by\_name, 11  
get\_condition\_biased\_comigrations, 12  
get\_enrichment, 29  
get\_layout, 13  
get\_mTOR\_proteins, 14  
get\_random\_data, 14  
get\_stats, 15  
get\_theme\_blank, 15  
get\_unknown\_ppi\_nodes, 16  
gn, 16  
gn\_desc, 17  
  
hprd\_ppi, 17  
hub\_score, 30, 33, 35  
  
layout\_, 31, 33, 35  
layout\_as\_star, 31, 33, 35  
layout\_as\_tree, 31, 33, 35  
layout\_in\_circle, 31, 33, 35  
layout\_nicely, 31, 33, 35  
layout\_on\_sphere, 31, 33, 35  
layout\_randomly, 31, 33, 35  
layout\_with\_gem, 31, 33, 35  
length2, 17  
load\_previous\_results, 18  
  
make\_random\_xina\_data, 19  
mclustModelNames, 28  
mutate\_colors, 20  
  
organize\_clusters, 21  
  
p.adjust, 4  
plot\_clusters, 21  
plot\_clusters\_all, 22  
plot\_condition\_compositions, 12, 20, 23  
plot\_enrichment\_results, 24  
plot\_NA, 25  
  
rank\_centrality, 25  
Round, 35  
  
scale, 28  
shapes, 26, 31, 33, 35  
string\_example, 26  
  
xina\_analysis, 16, 26, 30, 32, 34  
xina\_clustering, 4, 7, 9, 11, 12, 16, 20–23,  
26, 27, 27, 30, 32, 36  
xina\_enrichment, 24, 29  
xina\_plot\_all, 30  
xina\_plot\_bycluster, 32  
xina\_plot\_single, 34  
xina\_result\_example, 36