

# Package ‘crosstalkr’

May 18, 2023

**Title** Analysis of Graph-Structured Data with a Focus on  
Protein-Protein Interaction Networks

**Version** 1.0.2

**Description** Provides a general toolkit for drug target identification. We include functionality to reduce large graphs to subgraphs and prioritize nodes. In addition to being optimized for use with generic graphs, we also provides support to analyze protein-protein interactions networks from online repositories. For more details on core method, refer to Weaver et al. (2021) <<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008755>>.

**License** GPL (>= 3)

**biocViews**

**Imports** rlang, magrittr, withr, readr, dplyr, stringr, tidyr, tibble,  
igraph (>= 1.2.0), Matrix, ensemblDb, foreach, doParallel,  
Rcpp, iterators, ggplot2, STRINGdb

**LinkingTo** Rcpp

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Suggests** tidygraph, ggraph, testthat (>= 2.0.0), knitr,  
EnsDb.Hsapiens.v86, rmarkdown, here, msigdb

**Config/testthat/edition** 2

**VignetteBuilder** knitr

**Depends** R (>= 2.10)

**NeedsCompilation** yes

**Author** Davis Weaver [aut, cre] (0000-0003-3086-497X)

**Maintainer** Davis Weaver <davis.weaver@case.edu>

**Repository** CRAN

**Date/Publication** 2023-05-18 17:50:02 UTC

**R topics documented:**

add_expression . . . . .	3
add_value . . . . .	3
as_gene_symbol . . . . .	4
bootstrap_null . . . . .	4
calc_dnp_i . . . . .	6
calc_np . . . . .	6
calc_np_all . . . . .	7
calc_np_all_legacy . . . . .	7
calc_np_i . . . . .	8
check_crosstalk . . . . .	9
combine_null . . . . .	9
compute_crosstalk . . . . .	10
compute_dnp . . . . .	12
compute_np . . . . .	12
compute_null_dnp . . . . .	13
crosstalkr . . . . .	14
crosstalk_subgraph . . . . .	15
detect_inputtype . . . . .	16
dist_calc . . . . .	16
ensembl_type . . . . .	17
experiment_breakout . . . . .	17
fcalc_np_all . . . . .	18
final_combine . . . . .	18
final_dist_calc . . . . .	19
get_neighbors . . . . .	19
get_random_graph . . . . .	20
get_topn . . . . .	20
gfilter . . . . .	21
gfilter.ct . . . . .	22
gfilter.igraph_method . . . . .	22
gfilter.np . . . . .	23
gfilter.value . . . . .	23
is_ensembl . . . . .	24
is_entrez . . . . .	25
load_ppi . . . . .	25
match_seeds . . . . .	26
node_repression . . . . .	26
norm_colsum . . . . .	27
plot_ct . . . . .	28
ppi_intersection . . . . .	28
ppi_union . . . . .	29
prep_biogrid . . . . .	30
prep_stringdb . . . . .	30
sparseRWR . . . . .	31
supported_species . . . . .	32
tidy_expression . . . . .	32

<code>add_expression</code>	3
<code>to_taxon_id</code> . . . . .	33

**Index** **34**

<code>add_expression</code>	<i>attach expression values from user-provided expression vector to graph.</i>
-----------------------------	--

**Description**

attach expression values from user-provided expression vector to graph.

**Usage**

```
add_expression(exp, g)
```

**Arguments**

<code>exp</code>	expression vector - assumed to be a named vector where the values are expression and the names are the gene name
<code>g</code>	igraph object - will be filtered so that only nodes found in both <code>exp</code> and <code>g</code> are kept

**Value**

subgraph of `g` containing only shared keys with `exp` and with expression attached.

<code>add_value</code>	<i>Attach a generic user-provided value to graph</i>
------------------------	--

**Description**

Attach a generic user-provided value to graph

**Usage**

```
add_value(val, g, val_name = "value")
```

**Arguments**

<code>val</code>	named numeric vector where the names correspond to vertices in <code>g</code>
<code>g</code>	igraph object - will be filtered so that only nodes found in both <code>exp</code> and <code>g</code> are kept
<code>val_name</code>	str key for <code>val</code>

**Value**

subgraph of `g` containing only shared keys with `val` and `val` attached

as\_gene\_symbol                      *Convert from most other representations of gene name to gene.symbol*

---

**Description**

Convert from most other representations of gene name to gene.symbol

**Usage**

```
as_gene_symbol(x, edb = NULL)
```

**Arguments**

x	vector of ensemble.gene ids, ensemble.peptide ids, ensemble.transcript ids or entrez gene ids
edb	ensemble database object

**Value**

vector of gene symbols

**Examples**

```
#1) from numeric formatted entrez id
as_gene_symbol(1956)
#2) from character formatted entrez id
as_gene_symbol("1956")
#3) from ensemble gene id
as_gene_symbol("ENSG00000146648")
#4) From a vector of entrez ids
as_gene_symbol(c("123", "1956", "2012"))
```

---

bootstrap\_null                      *Bootstrap null distribution for RWR*

---

**Description**

This function will generate a bootstrapped null distribution to identify significant vertices in a PPI given a set of user-defined seed proteins. Bootstrapping is done by performing random walk with repeats repeatedly over "random" sets of seed proteins. Degree distribution of user-provided seeds is used to inform sampling.

**Usage**

```
bootstrap_null(  
  seed_proteins,  
  g,  
  n = 1000,  
  agg_int = 100,  
  gamma = 0.6,  
  eps = 1e-10,  
  tmax = 1000,  
  norm = TRUE,  
  set_seed = NULL,  
  cache = NULL,  
  seed_name = NULL,  
  ncores = 1  
)
```

**Arguments**

seed_proteins	user defined seed proteins
g	igraph object
n	number of random walks with repeats to create null distribution
agg_int	number of runs before we need to aggregate the results - necessary to save memory. set at lower numbers to save even more memory.
gamma	restart probability
eps	maximum allowed difference between the computed probabilities at the steady state
tmax	the maximum number of iterations for the RWR
norm	if True, w is normalized by dividing each value by the column sum.
set_seed	integer to set random number seed - for reproducibility
cache	A filepath to a folder downloaded files should be stored
seed_name	Name to give the cached ngull distribution - must be a character string
ncores	Number of cores to use - defaults to 1. Significant speedup can be achieved by using multiple cores for computation.

**Value**

data frame containing mean/ standard deviation for null distribution

**Examples**

```
#g <- prep_biogrid()  
#bootstrap_null(seed_proteins = c("EGFR", "KRAS"), g= g, ncores = 1, n = 10)
```

---

calc\_dnp\_i                    *helper function to calculate dnp for one sample*

---

**Description**

helper function to calculate dnp for one sample

**Usage**

```
calc_dnp_i(df, g, v_rm = NULL, keep_all = TRUE)
```

**Arguments**

df	dataframe with one cell line + log expression
g	igraph object containing ppi info
v_rm	passed to <a href="#">node_repression()</a>
keep_all	logical flag denoting if we should keep genes that we didn't calculate dnp for

**Value**

same dataframe with dnp calculated for each gene.

---

calc\_np                    *calculate network potential for one node.*

---

**Description**

calculate network potential for one node.

**Usage**

```
calc_np(c_i, c_j)
```

**Arguments**

c_i	expression for a given node.
c_j	vector of expressions for each neighbor of c_i

---

calc_np_all	<i>function to calculate the network potential for each protein in a user-provided vector - cpp internal version</i>
-------------	--

---

**Description**

function to calculate the network potential for each protein in a user-provided vector - cpp internal version

**Usage**

```
calc_np_all(exp, g, v = "default", neighbors = NULL)
```

**Arguments**

exp	expression vector - assumed to be a named vector where the values are expression and the names are the gene name
g	igraph object - will be filtered so that only nodes found in both exp and g are kept
v	character vector of nodes over which to calculate network potential.
neighbors	named list containing the neighbors for each node of graph g. If not provided, it will be computed

**Value**

dataframe containing network potential for each of the inputted gene names.

---

calc_np_all_legacy	<i>function to calculate the network potential for each protein in a user-provided vector</i>
--------------------	---

---

**Description**

Mostly just used to help debug the CPP version - not exported

**Usage**

```
calc_np_all_legacy(
  exp,
  g,
  v = as.character(names(igraph::V(g))),
  neighbors = NULL
)
```

**Arguments**

exp	expression vector - assumed to be a named vector where the values are expression and the names are the gene name
g	igraph object - will be filtered so that only nodes found in both exp and g are kept
v	character vector of nodes over which to calculate network potential.
neighbors	named list containing the neighbors for each node of graph g. If not provided, it will be computed

**Value**

dataframe containing network potential for each of the inputted gene names.

---

`calc_np_i`*helper function to calculate np for one sample*

---

**Description**

helper function to calculate np for one sample

**Usage**

```
calc_np_i(df, g)
```

**Arguments**

df	dataframe with one cell line + log expression
g	igraph object containing ppi info

**Value**

same dataframe with np calculated for each gene.



---

check_crosstalk	<i>Check to make sure incoming object is a valid crosstalk df.</i>
-----------------	--

---

**Description**

This function is a helper function for plot\_ct that verifies the input is a valid output of compute\_crosstalk

**Usage**

```
check_crosstalk(crosstalk_df)
```

**Arguments**

crosstalk\_df    a dataframe containing the results of compute\_crosstalk

**Value**

message if not correct object type, null otherwise

---

combine_null	<i>.combine function for compute_null foreach looping structure</i>
--------------	---

---

**Description**

.combine function for compute\_null foreach looping structure

**Usage**

```
combine_null(x)
```

**Arguments**

x                    aggregated data structure

**Value**

data.frame

---

compute_crosstalk	<i>Identify proteins with a statistically significant relationship to user-provided seeds.</i>
-------------------	--

---

### Description

compute\_crosstalk returns a dataframe of proteins that are significantly associated with user-defined seed proteins. These identified "crosstalkers" can be combined with the user-defined seed proteins to identify functionally relevant subnetworks. Affinity scores for every protein in the network are calculated using a random-walk with repeats (sparseRWR). Significance is determined by comparing these affinity scores to a bootstrapped null distribution (see bootstrap\_null). If using non-human PPI from string, refer to the stringdb documentation for how to specify proteins

### Usage

```
compute_crosstalk(  
  seed_proteins,  
  g = NULL,  
  use_ppi = TRUE,  
  ppi = "stringdb",  
  species = "homo sapiens",  
  n = 1000,  
  union = FALSE,  
  intersection = FALSE,  
  gamma = 0.6,  
  eps = 1e-10,  
  tmax = 1000,  
  norm = TRUE,  
  set_seed,  
  cache = NULL,  
  min_score = 700,  
  seed_name = NULL,  
  ncores = 1,  
  significance_level = 0.95,  
  p_adjust = "bonferroni",  
  agg_int = 100,  
  return_g = FALSE  
)
```

### Arguments

seed_proteins	user defined seed proteins
g	igraph network object.
use_ppi	bool, should g be a protein-protein interaction network? If false, user must provide an igraph object in g

ppi	character string describing the ppi to use: currently only "stringdb" and "biogrid" are supported.
species	character string describing the species of interest. For a list of supported species, see supported_species. Non human species are only compatible with "stringdb"
n	number of random walks with repeats to create null distribution
union	bool, should we take the union of string db and biogrid to compute the PPI? Only applicable for the human PPI
intersection	bool, should we take the intersection of string db and biogrid to compute the PPI? Only applicable for the human PPI
gamma	restart probability
eps	maximum allowed difference between the computed probabilities at the steady state
tmax	the maximum number of iterations for the RWR
norm	if True, w is normalized by dividing each value by the column sum.
set_seed	integer to set random number seed - for reproducibility
cache	A filepath to a folder downloaded files should be stored
min_score	minimum connectivity score for each edge in the network.
seed_name	Name to give the cached ngull distribution - must be a character string
ncores	Number of cores to use - defaults to 1. Significant speedup can be achieved by using multiple cores for computation.
significance_level	user-defined significance level for hypothesis testing
p_adjust	adjustment method to correct for multiple hypothesis testing: defaults to "holm". see p.adjust.methods for other potential adjustment methods.
agg_int	number of runs before we need to aggregate the results - necessary to save memory. set at lower numbers to save even more memory.
return_g	bool, should we return the graph used? mostly for internal use

**Value**

data frame containing affinity score, p-value, for all "crosstalkers" related to a given set of seeds

**Examples**

```
#1) easy to use for querying biological networks - n = 10000 is more appropriate for actual analyses
#compute_crosstalk(c("EGFR", "KRAS"), n = 10)
```

```
#2) Also works for any other kind of graph- just specify g (must be igraph formatted as of now)
g <- igraph::sample_gnp(n = 1000, p = 10/1000)
compute_crosstalk(c(1,3,5,8,10), g = g, use_ppi = FALSE, n = 100)
```

---

compute_dnp	<i>main function to compute delta np for every gene in a given dataframe - assumes compute_np has already been run for a given dataset</i>
-------------	--

---

### Description

This function takes a tidy dataframe as input containing RNA sequencing data for one or more samples and conducts in-silico repression. Make sure to run with the same arguments for ppi and cache to maintain consistency for a given pipeline.

### Usage

```
compute_dnp(
  cache = NULL,
  df,
  experiment_name,
  ppi,
  ncores = 1,
  min_score = NULL
)
```

### Arguments

cache	user-provided filepath for where to store data etc
df	dataframe output of <a href="#">compute_np</a>
experiment_name	name of the experiment for saving output.
ppi	should we use biogrid or stringdb for the PPI
ncores	number of cores to use for calculations
min_score	if ppi is stringdb, which minimum score should we use to filter edges?

### Value

data.frame

---

compute_np	<i>main function to compute np from a user-provided expression matrix.</i>
------------	--

---

### Description

main function to compute np from a user-provided expression matrix.

**Usage**

```
compute_np(
  cache = NULL,
  experiment_name,
  ppi = "biogrid",
  min_score = NULL,
  exp_mat,
  mir_paper = TRUE,
  ncores = 1
)
```

**Arguments**

cache	user-provided filepath for where to store data etc
experiment_name	name of the experiment for saving output.
ppi	should we use biogrid or stringdb for the PPI
min_score	if ppi is stringdb, which minimum score should we use to filter edges?
exp_mat	expression matrix where columns are samples and rows are features
mir_paper	are we running this in the context of the mir paper? a few quirks of that data
ncores	number of cores to use for calculations

**Value**

tidy data frame with one column for expression and another for np

---

compute\_null\_dnp      *function to compute null distribution of dnp*

---

**Description**

compute\_null\_dnp calculates a null distribution for the change in network potential for for each node in a cell signaling network.

**Usage**

```
compute_null_dnp(
  cache = NULL,
  df,
  ppi = "biogrid",
  n,
  n_genes = 50,
  experiment_name,
  ncores = 4,
  min_score = NULL
)
```

### Arguments

cache	user-provided filepath for where to store data etc
df	output of <code>compute_dnp()</code>
ppi	should we use biogrid or stringdb for the PPI
n	number of permutations
n_genes	integer describing number of genes per sample that we will compute the null distribution for
experiment_name	name of the experiment for saving output.
ncores	number of cores to use for calculations
min_score	if ppi is stringdb, which minimum score should we use to filter edges?

### Details

The input for this function will be the output of `compute_dnp()`. To compute the null distribution, the nodes in the provided cell signaling network will be randomly permuted `n` times, with `dnp` computed on each new cell signaling network. The mean and standard error of `dnp` for this set of random networks will constitute the null model that we will use for comparison. Be warned that this operation is extremely expensive computationally. It is recommended to either use a high-performance cluster or limit the computation of the null distribution to a small number of nodes. To distribute the workload over multiple cores, just specify `ncores`.

### Value

df, also saves to cache if specified

### See Also

`compute_dnp()` and `compute_np()`

---

crosstalkr	<i>crosstalkr: A package for the identification of functionally relevant subnetworks from high-dimensional omics data.</i>
------------	--

---

### Description

crosstalkr provides a key user function, `compute_crosstalk` as well as several additional functions that assist in setup and visualization (under development).

**crosstalkr functions**

`compute_crosstalk` calculates affinity scores of all proteins in a network relative to user-provided seed proteins. Users can use the human interactome or provide a network represented as an `igraph` object.

`sparseRWR` performs random walk with restarts on a sparse matrix. Compared to dense matrix implementations, this should be extremely fast.

`bootstrap_null` Generates a null distribution based on `n` calls to `sparseRWR`

`setup_init` manages download and storage of interactome data to speed up future analysis

`plot_ct` allows users to visualize the subnetwork identified in `compute_crosstalk`. This function relies on the `ggraph` framework. Users are encouraged to use `ggraph` or other network visualization packages for more customized figures.

`crosstalk_subgraph` converts the output of `compute_crosstalk` to a `tidygraph` object containing only the identified nodes and their connections to the user-provided `seed_proteins`. This function also adds `degree`, `degree_rank`, and `seed_label` as attributes to the identified subgraph to assist in plotting.

---

<code>crosstalk_subgraph</code>	<i>Helper function to generate subgraph from crosstalk_df output of compute_crosstalk</i>
---------------------------------	---

---

**Description**

Useful if the user wants to carry out further analysis or design custom visualizations.

**Usage**

```
crosstalk_subgraph(crosstalk_df, g, seed_proteins, tg = TRUE)
```

**Arguments**

<code>crosstalk_df</code>	a dataframe containing the results of <code>compute_crosstalk</code>
<code>g</code>	<code>igraph</code> network object.
<code>seed_proteins</code>	user defined seed proteins
<code>tg</code>	bool do we want to tidy the graph for plotting?

**Value**

a `tidygraph` structure containing information about the `crosstalkr` subgraph

**Examples**

```
## Not run:
ct_df <- compute_crosstalk(c("EGFR", "KRAS"))
g <- prep_biogrid()
crosstalk_subgraph(ct_df, g = g, seed_proteins = c("EGFR", "KRAS"))

## End(Not run)
```

---

detect_inputtype	<i>Determine which format of gene is used to specify by user-defined seed proteins</i>
------------------	--

---

**Description**

Determine which format of gene is used to specify by user-defined seed proteins

**Usage**

```
detect_inputtype(x)
```

**Arguments**

x                    vector of gene symbols

**Value**

"gene\_symbol", "entrez\_id", "ensemble\_id" or "other"

---

dist_calc	<i>Internal function that computes the mean/stdev for each gene from a wide-format data frame.</i>
-----------	--

---

**Description**

This function is called by the high-level function "bootstrap\_null". Not expected to be used by end-users - we only export it so that environments inside foreach loops can find it.

**Usage**

```
dist_calc(df, seed_proteins)
```

**Arguments**

df                    : numeric vector  
seed\_proteins        user defined seed proteins

**Value**

data.frame containing summary statistics for the computed null distribution



---

ensembl_type	<i>Determine if ensembl id is a Protein, gene, or transcript_id</i>
--------------	---

---

**Description**

Determine if ensembl id is a Protein, gene, or transcript\_id

**Usage**

```
ensembl_type(x)
```

**Arguments**

x                      vector or single gene symbol

**Value**

character: "PROTEINID", "GENEID", "TRANSCRIPTID"

---

experiment_breakout	<i>helper function to split experiment names into constituent parts</i>
---------------------	---

---

**Description**

this is highly specific to the miR paper

**Usage**

```
experiment_breakout(df)
```

**Arguments**

df                      dataframe

**Value**

data.frame

---

fcalc_np_all	<i>Function to calculate the network potential for vertices v</i>
--------------	---

---

**Description**

Function to calculate the network potential for vertices v

**Usage**

```
fcalc_np_all(neighbors, vertices, v, exp)
```

**Arguments**

neighbors	list of neighbors for every node in the graph, type Rcpp::list
vertices	node list for graph, type Rcpp::StringVector
v	list of nodes for which we plan to calculate network potential
exp	named vector of expression for each node in vertices

---

final_combine	<i>final .combine function to run in compute_null_dnp foreach looping structure</i>
---------------	---

---

**Description**

final .combine function to run in compute\_null\_dnp foreach looping structure

**Usage**

```
final_combine(x)
```

**Arguments**

x	aggregated info
---	-----------------

**Value**

data.frame

---

final_dist_calc	<i>Internal function that computes the mean/stdev for each gene from a wide-format data frame.</i>
-----------------	--

---

**Description**

This function is called by the high-level function "bootstrap\_null".

**Usage**

```
final_dist_calc(df_list)
```

**Arguments**

df\_list : list of dataframes from foreach loop in bootstrap\_null

**Value**

data.frame

---

get_neighbors	<i>function to get graph neighbors (along with their expression values) for a given gene in a given network g</i>
---------------	---

---

**Description**

just a wrapper around `igraph::neighbors()` for convenience

**Usage**

```
get_neighbors(gene, g)
```

**Arguments**

gene gene to grab neighbors from.  
g igraph object - will be filtered so that only nodes found in both exp and g are kept

**Value**

named numeric vector.

---

get_random_graph	<i>Helper function for compute_null_dnp - returns a graph with randomly permuted edges.</i>
------------------	---

---

**Description**

currently just a wrapper for `igraph::rewire` but may add more functionality in the future

**Usage**

```
get_random_graph(g)
```

**Arguments**

<code>g</code>	graph to be permuted
----------------	----------------------

**Value**

igraph

**See Also**

[igraph::rewire\(\)](#)

---

get_topn	<i>Helper function for compute_null_dnp - returns the top n genes by dnp for each sample</i>
----------	--

---

**Description**

Helper function for `compute_null_dnp` - returns the top n genes by dnp for each sample

**Usage**

```
get_topn(df, n_genes)
```

**Arguments**

<code>df</code>	output of <a href="#">compute_dnp()</a>
<code>n_genes</code>	integer describing number of genes per sample that we will compute the null distribution for

---

`gfilter`*Generic function to filter either an igraph object or a PPI network*

---

**Description**

Generic function to filter either an igraph object or a PPI network

**Usage**

```
gfilter(  
  method = NULL,  
  g = NULL,  
  val = NULL,  
  use_ppi,  
  igraph_method = NULL,  
  n = 100,  
  desc = TRUE,  
  ...  
)
```

**Arguments**

<code>method</code>	str
<code>g</code>	igraph object
<code>val</code>	named numeric vector - some measure of node state (i.e. gene expression in the case of a PPI)
<code>use_ppi</code>	bool - should we use a ppi from online repository?
<code>igraph_method</code>	bool - is the user-provided method an igraph node scoring function?
<code>n</code>	int - number of nodes to include in the returned subgraph
<code>desc</code>	bool - do we want the top or bottom examples of the provided metric
<code>...</code>	additional params passed to <code>load_ppi()</code> or <code>compute_crosstalk()</code>

**Value**

igraph

**See Also**

[gfilter.ct](#), [gfilter.np](#), [gfilter.igraph\\_method](#)

---

gfilter.ct	<i>Method to filter the graph based on parameters passed to compute_crosstalk</i>
------------	---

---

**Description**

Method to filter the graph based on parameters passed to compute\_crosstalk

**Usage**

```
gfilter.ct(seeds, return_df = FALSE, ...)
```

**Arguments**

seeds	vector (str or numeric) user provided vertex ids to use as seeds in the random walk with restarts'
return_df	bool should we return a list containing the filtered graph + the RWR output that was used to do the filtering?
...	additional arguments passed to <a href="#">compute_crosstalk()</a>

**Value**

igraph object

---

gfilter.igraph_method	<i>Method to filter graph based on any igraph method that scores vertices.</i>
-----------------------	--

---

**Description**

Method to filter graph based on any igraph method that scores vertices.

**Usage**

```
gfilter.igraph_method(g, use_ppi = TRUE, method, n = 500, desc, val_name, ...)
```

**Arguments**

g	igraph object
use_ppi	bool - should we use a ppi from online repository?
method	str
n	int - number of nodes to include in the returned subgraph
desc	bool - do we want the top or bottom examples of the provided metric
val_name	str
...	additional parameters passed to <a href="#">load_ppi</a>

**Value**

igraph

---

gfilter.np	<i>Method to filter graph based on network potential values.</i>
------------	--

---

**Description**

convenience function - it just calls gfilter.value after computing np

**Usage**

```
gfilter.np(g, val, use_ppi = TRUE, n = 500, desc, ...)
```

**Arguments**

g	igraph object
val	named numeric vector - some measure of node state (i.e. gene expression in the case of a PPI)
use_ppi	bool - should we use a ppi from online repository?
n	int - number of nodes to include in the returned subgraph
desc	bool - do we want the top or bottom examples of the provided metric
...	additional params passed to <a href="#">load_ppi()</a> or <a href="#">compute_crosstalk()</a>

**Details**

For more information on network potential, see [related paper](#)

**Value**

igraph

---

gfilter.value	<i>Method to filter graph based on user provided value</i>
---------------	--

---

**Description**

Method to filter graph based on user provided value

**Usage**

```
gfilter.value(g, val, use_ppi = TRUE, n = 500, val_name = "value", desc, ...)
```

**Arguments**

g	igraph object
val	named numeric vector - some measure of node state (i.e. gene expression in the case of a PPI)
use_ppi	bool - should we use a ppi from online repository?
n	int - number of nodes to include in the returned subgraph
val_name	str
desc	bool - do we want the top or bottom examples of the provided metric
...	additional params passed to <a href="#">load_ppi()</a> or <a href="#">compute_crosstalk()</a>

**Value**

igraph

---

is_ensembl	<i>Determine if a character vector contains ensembl gene_ids</i>
------------	--

---

**Description**

Determine if a character vector contains ensembl gene\_ids

**Usage**

```
is_ensembl(x)
```

**Arguments**

x	vector or single gene symbol
---	------------------------------

**Value**

logical



---

is_entrez	<i>Determine if a character vector contains entrez gene_ids</i>
-----------	---

---

**Description**

Determine if a character vector contains entrez gene\_ids

**Usage**

```
is_entrez(x)
```

**Arguments**

x	vector or single gene symbol
---	------------------------------

**Value**

logical

---

load_ppi	<i>Helper function to load requested PPI w/ parameters</i>
----------	--

---

**Description**

Helper function to load requested PPI w/ parameters

**Usage**

```
load_ppi(
  cache = NULL,
  union = FALSE,
  intersection = FALSE,
  species = "9606",
  min_score = 0,
  ppi = "stringdb"
)
```

**Arguments**

cache	A filepath to a folder downloaded files should be stored
union	bool
intersection	bool
species	species code either using latin species name or taxon id
min_score	minimum connectivity score for each edge in the network.
ppi	str

**Value**

igraph object

---

match_seeds	<i>Identify random sets of seeds with similar degree distribution to parent seed proteins</i>
-------------	---

---

**Description**

This function will generate n character vectors of seeds to be passed to sparseRWR as part of the construction of a bootstrapped null distribution for significance testing.

**Usage**

```
match_seeds(g, seed_proteins, n, set_seed = NULL)
```

**Arguments**

g	igraph object representing the network under study. specified by "ppi" in bootstrap_null
seed_proteins	user defined seed proteins
n	number of random walks with repeats to create null distribution
set_seed	integer to set random number seed - for reproducibility

**Value**

list of character vectors: randomly generated seed proteins with a similar degree distribution to parent seed proteins

---

node_repression	<i>Function to eliminate a node from a network g and calculate the change in some measure of network state</i>
-----------------	--

---

**Description**

this function is still under development.

**Usage**

```
node_repression(
  g,
  v_rm = as.character(names(igraph::V(g))),
  exp,
  state_function = calc_np_all,
  neighbors_only = TRUE,
  ...
)
```

**Arguments**

<code>g</code>	igraph network object
<code>v_rm</code>	index of vertices to remove
<code>exp</code>	expression vector for nodes in graph <code>g</code>
<code>state_function</code>	function to use to calculate network state before and after node_repression
<code>neighbors_only</code>	logical designating whether state function should be calculated for all nodes or just neighbors
<code>...</code>	additional parameters passed to state function.

---

<code>norm_colsum</code>	<i>Function to normalize adjacency matrix by dividing each value by the colsum.</i>
--------------------------	---

---

**Description**

Function to normalize adjacency matrix by dividing each value by the colsum.

**Usage**

```
norm_colsum(w)
```

**Arguments**

<code>w</code>	The adjacency matrix of a given graph in sparse format - dgCMatrix
----------------	--

**Value**

input matrix, normalized by column sums

**Examples**

```
# 1) Normalize by column sum on a simple matrix
v1 = c(1,1,1,0)
v2 = c(0,0,0,1)
v3 = c(1,1,1,0)
v4 = c(0,0,0,1)
w = matrix(data = c(v1,v2,v3,v4), ncol = 4, nrow = 4)
norm_colsum(w)
```

---

plot_ct	<i>Plot subnetwork identified using the compute_crosstalk function</i>
---------	--

---

### Description

Convenience function for plotting crosstalkers - if you want to make more customized/dynamic figures, there are lots of packages that can facilitate that, including: visnetwork, ggraph, and even the base R plotting library

### Usage

```
plot_ct(crosstalk_df, g, label_prop = 0.1, prop_keep = 0.4)
```

### Arguments

crosstalk_df	a dataframe containing the results of compute_crosstalk
g	igraph network object.
label_prop	Proportion of nodes to label - based on degree
prop_keep	How many proteins do we want to keep in the visualization (as a proportion of total) - subsets on top x proteins ranked by affinity score

### Value

NULL, draws the identified subgraph to device\

### Examples

```
## Not run:
ct_df <- compute_crosstalk(c("EGFR", "KRAS"))
g <- prep_biogrid()
plot_ct(ct_df, g = g)

## End(Not run)
```

---

ppi_intersection	<i>Function to allow users to choose the intersection of stringdb and biogrid Only works with the human PPI. min_score parameter only applies to strindb</i>
------------------	--

---

### Description

Function to allow users to choose the intersection of stringdb and biogrid Only works with the human PPI. min\_score parameter only applies to strindb

**Usage**

```
ppi_intersection(cache = NULL, min_score = 800, edb = "default")
```

**Arguments**

cache	A filepath to a folder downloaded files should be stored
min_score	minimum connectivity score for each edge in the network.
edb	ensemble database object

**Value**

igraph object corresponding to PPI following intersection

---

ppi_union	<i>Function to allow users to choose the union of stringdb and biogrid Only works with the human PPI. min_score parameter only applies to strindb</i>
-----------	---

---

**Description**

Function to allow users to choose the union of stringdb and biogrid Only works with the human PPI. min\_score parameter only applies to strindb

**Usage**

```
ppi_union(cache = NULL, min_score = 0, edb = "default")
```

**Arguments**

cache	A filepath to a folder downloaded files should be stored
min_score	minimum connectivity score for each edge in the network.
edb	ensemble database object

**Value**

igraph object corresponding to PPI following union

---

```
prep_biogrid          Prepare biogrid for use in analyses
```

---

**Description**

Prepare biogrid for use in analyses

**Usage**

```
prep_biogrid(cache = NULL)
```

**Arguments**

cache                A filepath to a folder downloaded files should be stored

**Value**

igraph object built from the adjacency matrix downloaded from thebiogrid.org.

---

```
prep_stringdb        Prepare Stringdb for use in analyses
```

---

**Description**

Basically a wrapper around the `get_graph` method from the `stringdb` package

**Usage**

```
prep_stringdb(
  cache = NULL,
  edb = "default",
  min_score = 200,
  version = "11.5",
  species = "homo sapiens"
)
```

**Arguments**

cache                A filepath to a folder downloaded files should be stored  
 edb                  ensemble database object  
 min\_score            minimum connectivity score for each edge in the network.  
 version              stringdb version  
 species              species code either using latin species name or taxon id

**Value**

igraph object built from the adjacency matrix downloaded from stringdb.

---

sparseRWR	<i>Perform random walk with repeats on a sparse matrix</i>
-----------	--

---

**Description**

This function borrows heavily from the RWR function in the RANKS package (cite here)

**Usage**

```
sparseRWR(seed_proteins, w, gamma = 0.6, eps = 1e-10, tmax = 1000, norm = TRUE)
```

**Arguments**

seed_proteins	user defined seed proteins
w	The adjacency matrix of a given graph in sparse format - dgCMatrix
gamma	restart probability
eps	maximum allowed difference between the computed probabilities at the steady state
tmax	the maximum number of iterations for the RWR
norm	if True, w is normalized by dividing each value by the column sum.

**Value**

numeric vector, affinity scores for all nodes in graph relative to provided seeds

**Examples**

```
# 1) Run Random walk with restarts on a simple matrix
v1 = c(1,1,1,0)
v2 = c(0,0,0,1)
v3 = c(1,1,1,0)
v4 = c(0,0,0,1)
w = matrix(data = c(v1,v2,v3,v4), ncol = 4, nrow = 4)
sparseRWR(seed_proteins = c(1,3), w = w, norm = TRUE)

# 2) Works just as well on a sparse matrix
v1 = c(1,1,1,0)
v2 = c(0,0,0,1)
v3 = c(1,1,1,0)
v4 = c(0,0,0,1)
w = matrix(data = c(v1,v2,v3,v4), ncol = 4, nrow = 4)
w = Matrix::Matrix(w, sparse = TRUE)
sparseRWR(seed_proteins = c(1,4), w = w, norm = TRUE)

#3) Sample workflow for use with human protein-protein interaction network
#g <- prep_biogrid()
#w <- igraph::as_adjacency_matrix(g)
#sparseRWR(seed_proteins = c("EGFR", "KRAS"), w = w, norm = TRUE)
```

---

supported_species	<i>returns a dataframe with information on supported species</i>
-------------------	--

---

**Description**

returns a dataframe with information on supported species

**Usage**

```
supported_species()
```

**Value**

dataframe

---

tidy_expression	<i>helper function to convert expression matrix to tidy dataframe (if not already)</i>
-----------------	--

---

**Description**

helper function to convert expression matrix to tidy dataframe (if not already)

**Usage**

```
tidy_expression(df)
```

**Arguments**

df	dataframe
----	-----------

**Value**

data.frame



---

to_taxon_id	<i>helper to convert user-inputs to ncbi reference taxonomy.</i>
-------------	--

---

**Description**

helper to convert user-inputs to ncbi reference taxonomy.

**Usage**

to\_taxon\_id(species)

**Arguments**

species            user-inputted species

**Value**

string corresponding to taxon id

# Index

add\_expression, 3  
add\_value, 3  
as\_gene\_symbol, 4  
  
bootstrap\_null, 4  
  
calc\_dnp\_i, 6  
calc\_np, 6  
calc\_np\_all, 7  
calc\_np\_all\_legacy, 7  
calc\_np\_i, 8  
check\_crosstalk, 9  
combine\_null, 9  
compute\_crosstalk, 10  
compute\_crosstalk(), 21–24  
compute\_dnp, 12  
compute\_dnp(), 14, 20  
compute\_np, 12, 12  
compute\_np(), 14  
compute\_null\_dnp, 13  
crosstalk\_subgraph, 15  
crosstalkr, 14  
  
detect\_inputtype, 16  
dist\_calc, 16  
  
ensembl\_type, 17  
experiment\_breakout, 17  
  
fcalc\_np\_all, 18  
final\_combine, 18  
final\_dist\_calc, 19  
  
get\_neighbors, 19  
get\_random\_graph, 20  
get\_topn, 20  
gfilter, 21  
gfilter.ct, 21, 22  
gfilter.igraph\_method, 21, 22  
gfilter.np, 21, 23  
gfilter.value, 23  
  
igraph::neighbors(), 19  
igraph::rewire(), 20  
is\_ensembl, 24  
is\_entrez, 25  
  
load\_ppi, 22, 25  
load\_ppi(), 21, 23, 24  
  
match\_seeds, 26  
  
node\_repression, 26  
node\_repression(), 6  
norm\_colsum, 27  
  
plot\_ct, 28  
ppi\_intersection, 28  
ppi\_union, 29  
prep\_biogrid, 30  
prep\_stringdb, 30  
  
sparseRWR, 31  
supported\_species, 32  
  
tidy\_expression, 32  
to\_taxon\_id, 33