Package ‘netkit’

July 24, 2025

Type Package

Title Network Analysis Toolkit for Biological Graphs

Version 0.0.1

Author Dr. Alex Gallinat [aut, cre]

Maintainer Alex Gallinat, PhD <alex.gaoc@gmail.com>

Description A set of tools for analyzing and visualizing biological and functional networks.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports igraph,
ineq,
dplyr,
expm,
ggplot2,
ggExtra,
ggrepel,
graphics,
grDevices,
tibble,
scales,
Matrix,
utils,
progress,
pracma,
rlang,
future,
future.apply,
RSpectra

Suggests testthat (>= 3.0.0),
knitr,
rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Contents

`assign_attributes` *Assign Vertex and Edge Attributes to an igraph Graph*

Description

Adds or updates vertex and edge attributes in an `igraph` object using user-provided metadata tables. Vertex attributes are matched by the first column of `nodes_table`, and edge attributes are matched using the first two columns of `edge_table`, taking graph direction into account. Only matching nodes and edges are updated. Warnings are issued when there are unmatched entries.

Usage

```
assign_attributes(
  graph,
  nodes_table = NULL,
  edge_table = NULL,
  overwrite = TRUE
)
```

Arguments

`graph`	An `igraph` object or a data frame containing a symbolic edge list.
`nodes_table`	Optional. A `data.frame` whose first column corresponds to vertex names.
`edge_table`	Optional. A `data.frame` whose first two columns correspond to source and target vertices.
`overwrite`	Logical. If `TRUE`, existing attributes are overwritten. If `FALSE`, existing attributes are preserved. Default is `TRUE`.

Value

An `igraph` object with added or updated attributes.

`calculate_roles` *Calculate Network Roles Based on Within-Module Z-Score and Participation Coefficient*

Description

Implements the node role classification system of Guimerà & Amaral (2005) by calculating the within-module degree z-score and the participation coefficient for each node in a network. Nodes are assigned to one of seven role categories (R1–R7) based on their local modular connectivity.

Usage

```
calculate_roles(
  graph,
  communities = NULL,
  cluster.method = "spinglass",
  plot = TRUE,
  highlight_roles = TRUE,
  hub_z = 2.5,
  label_region = NULL,
  label.size = 12
)
```

Arguments

`graph`	An `igraph` object representing the network.
`communities`	Optional. A community clustering object (as returned by an `igraph` clustering function), or a named membership vector. If `NULL`, community detection is performed using `cluster.method`.
`cluster.method`	Character. Clustering algorithm to use if `communities` is `NULL`. Default is `"spinglass"`. Passed to `find_modules()`.
`plot`	Logical. Whether to generate a 2D plot of participation coefficient (P) vs. within-module z-score (z). Default is `TRUE`.
`highlight_roles`	Logical. If `TRUE`, the role regions in the z–P plane are shaded for visual clarity. Default is `TRUE`.
`hub_z`	Numeric. Threshold for defining hubs in terms of within-module z-score. Default is `2.5`.
`label_region`	Optional character vector of role labels (e.g., `c("R4", "R7")`) indicating which role regions should have their nodes labeled in the plot. Default is `NULL`.
`label.size`	Numeric. Base font size for plot text. Default is `12`.

Details

If no community structure is provided, modules are automatically detected using the specified clustering method. The function can optionally produce a 2D role plot (z vs. P) highlighting the canonical role regions.

The node roles are defined as:

R1	Ultra-peripheral (non-hub): $z < 2.5, P <= 0.05$
R2	Peripheral (non-hub): $z < 2.5, 0.05 < P <= 0.6$
R3	Non-hub connector: $z < 2.5, 0.6 < P <= 0.8$
R4	Non-hub kinless: $z < 2.5, P > 0.8$
R5	Provincial hub: $z >= 2.5, P <= 0.3$
R6	Connector hub: $z >= 2.5, 0.3 < P <= 0.75$
R7	Kinless hub: $z >= 2.5, P > 0.75$

Value

A list with three elements:

`plot` The `ggplot2` object (only if `plot = TRUE`).

`roles_definitions` A data frame describing the seven role types and their conditions.

`result` A data frame with node-level information: node name, module, z-score, participation coefficient, and assigned role.

References

Guimerà, R., & Amaral, L. A. N. (2005). Functional cartography of complex metabolic networks. *Nature*, 433(7028), 895–900. doi:10.1038/nature03288

See Also

`find_modules()`, `igraph::cluster_spinglass()`, `igraph::membership()`

Examples

```
## Not run:
g <- igraph::sample_gnp(200, 0.05, directed = F)
igraph::V(g)$name <- as.character(1:200)
result <- calculate_roles(g, plot = TRUE)
head(result$result)

## End(Not run)
```

compare_networks *Compare Two Networks*

Description

This function compares two networks using summary metrics, degree distributions, and topological similarity measures. It overlays the complementary cumulative frequency distributions (CCDFs) of degree and returns a combined report.

Usage

```
compare_networks(
  graph1,
  graph2,
  remove_singles = FALSE,
  show_PL = TRUE,
  PL_exponents = c(2, 3),
  colors = c("#e41a1c", "#000831", "#9c52f2", "#b8b8ff"),
  label.size = 12
)
```

Arguments

`graph1`	An igraph object or data.frame (edge list).
`graph2`	An igraph object or data.frame (edge list).
`remove_singles`	Logical; remove single nodes before analysis.
`show_PL`	Logical; whether to fit and display power law exponents.
`PL_exponents`	Vector; power-law slopes to show.
`colors`	Optional vector of colors for the CCDF plot.
`label.size`	Labels' size in the CCDF plot.

Value

A list with:

- metrics1: Summary metrics for graph1
- metrics2: Summary metrics for graph2
- CCDF_plot: ggplot overlay of ccdfs
- jaccard_similarity: Jaccard index of edge sets
- node_overlap: Fraction of shared nodes
- edge_overlap: Fraction of shared edges
- ks_test: KS test result for degree distributions

Examples

```
## Not run:
library(igraph)
g1 <- sample_pa(100)
g2 <- sample_gnp(100, 0.05, directed = F)
compare_networks(g1, g2)

## End(Not run)
```

`find_bottlenecks` *Identify and Bottleneck Nodes in a Network*

Description

Identifies bottleneck nodes in an `igraph` network as those with low degree and high betweenness centrality. The function supports both standardized (z-score) and quantile-based thresholding. Optionally, it produces a 2D scatter plot with bottlenecks highlighted.

Usage

```
find_bottlenecks(
  graph,
  method = c("zscore", "quantile"),
  degree_threshold = -1,
  betweenness_threshold = 1,
  degree_quantile = 0.25,
  betweenness_quantile = 0.75,
  log_transform = TRUE,
  plot = TRUE,
  focus_color = "skyblue",
  bottleneck_names = TRUE,
  bottleneck_cex = 3,
  gg_extra = list()
)
```

Arguments

`graph`	An `igraph` object representing the network to analyze or a data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes.
`method`	Character. Method to define bottlenecks: `"zscore"` or `"quantile"`.
`degree_threshold`	Numeric. Upper threshold for standardized degree (only used if `method = "zscore"`).
`betweenness_threshold`	Numeric. Lower threshold for standardized betweenness (used in both methods).
`degree_quantile`	Numeric between 0 and 1. Quantile threshold for degree (used if `method = "quantile"`).
`betweenness_quantile`	Numeric between 0 and 1. Quantile threshold for betweenness (used if `method = "quantile"`).
`log_transform`	Logical. If `TRUE`, applies `log1p` transformation to degree and betweenness.
`plot`	Logical. If `TRUE`, generates a plot of degree vs. betweenness highlighting bottlenecks.
`focus_color`	Character. Color to display in the focus area of the plot (bottlenecks region).
`bottleneck_names`	Logical. If `TRUE`, labels bottleneck nodes on the plot.

`bottleneck_cex`	Numeric. Font size scaling for bottleneck labels on the plot.
`gg_extra`	List. Additional user-defined layers for the returned ggplot. eg. list(ylim(-2,2), theme_bw(), theme(legend.position = "none"))

Value

A list with the following components:

`method` A message describing the method and thresholds used.

`result` A `tibble` with node name, degree, betweenness, transformed metrics, and bottleneck status.

`graph` The original graph with a new vertex attribute `bottleneck` (logical).

If `plot` = `TRUE`, a scatter plot of degree vs. betweenness is displayed, highlighting bottlenecks.

Examples

```
## Not run:
library(igraph)
g <- sample_pa(100)
find_bottlenecks(g, method = "quantile", plot = TRUE)

## End(Not run)
```

`find_hubs` *Identify Hub Nodes in a Network*

Description

This function identifies hub nodes in an `igraph` network based on degree and betweenness centrality using either z-score or quantile thresholds. Optionally, it visualizes the classification using a scatter plot with marginal histograms.

Usage

```
find_hubs(
  graph,
  method = c("zscore", "quantile"),
  degree_threshold = 3,
  betweenness_threshold = 1,
  degree_quantile = 0.95,
  betweenness_quantile = 0.95,
  log_transform = TRUE,
  plot = TRUE,
  focus_color = "darkgreen",
  label.size = 12,
  hub_names = TRUE,
  hub_cex = 3,
  gg_extra = list()
)
```

Arguments

`graph`	An `igraph` object or a data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes.
`method`	Character. Method to identify hubs: `"zscore"` (standardized metrics) or `"quantile"` (empirical percentiles).
`degree_threshold`	Numeric. Threshold for standardized degree (only used if `method = "zscore"`).
`betweenness_threshold`	Numeric. Threshold for standardized betweenness (only used if `method = "zscore"`).
`degree_quantile`	Numeric between 0 and 1. Quantile threshold for degree (used if `method = "quantile"`).
`betweenness_quantile`	Numeric between 0 and 1. Quantile threshold for betweenness (used if `method = "quantile"`).
`log_transform`	Logical. If `TRUE`, applies log-transformation to degree and betweenness metrics.
`plot`	Logical. If `TRUE`, generates a plot of the degree vs. betweenness classification.
`focus_color`	Character. Color to display in the focus area of the plot (hubs region).
`label.size`	Numeric. Base font size for plot elements. Passed to `theme_classic`.
`hub_names`	Logical. If `TRUE`, adds node labels to identified hubs on the plot.
`hub_cex`	Numeric. Font size scaling factor for hub labels on the plot.
`gg_extra`	List. Additional user-defined layers for the returned ggplot. eg. list(ylim(-2,2), theme_bw(), theme(legend.position = "none"))

Value

A list with the following components:

`method` Description of the method and thresholds used.

`result` A `tibble` with node name, degree, betweenness, transformed metrics, and hub status.

`graph` The original graph with a new vertex attribute `is_hub`.

If `plot = TRUE`, a scatter plot of degree vs. betweenness is displayed with hub nodes highlighted.

Examples

```
## Not run:
library(igraph)
g <- sample_pa(100)
find_hubs(g, method = "quantile", plot = TRUE)

## End(Not run)
```

`find_modules` *Detect and Visualize Network Modules (Communities)*

Description

Identifies modules (communities) in a network using a variety of community detection algorithms from the **igraph** package. Optionally filters out small modules, visualizes the detected modules, and returns induced subgraphs for each module.

Usage

```
find_modules(
  graph,
  method = "louvain",
  min_size = 3,
  no.of.communities = NULL,
  return_subgraphs = FALSE,
  plot = TRUE,
  label = FALSE,
  ...
)
```

Arguments

`graph`	An `igraph` object representing the network to analyze or a data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes.
`method`	Character. Community detection method. Options include: `"louvain"`, `"walktrap"`, `"infomap"`, `"edge_betweenness"`, `"fluid_communities"`, `"fast_greedy"`, `"leading_eigen"`, `"leiden"`, and `"spinglass"`.
`min_size`	Integer. Minimum number of nodes required to retain a module. Modules smaller than this size are discarded. Default is 3.
`no.of.communities`	Integer. Required only when `method = "fluid_communities"`. Specifies the number of communities to find.
`return_subgraphs`	Logical. If `TRUE`, returns a list of induced subgraphs for each detected module.
`plot`	Logical. If `TRUE`, generates a network plot colored by module.
`label`	Logical. If `TRUE`, displays node labels in the plot.
`...`	Additional parameters passed to the `plot_Net()` function for customizing the plot.

Value

A list with the following components:

`module_table` A tibble mapping each node to its module assignment.

`n_modules` The number of modules that meet the `min_size` threshold.

`subgraphs` A named list of subgraphs for each module (only if `return_subgraphs = TRUE`).

method The community detection method used.

graph The input graph with assigned 'module' and 'color' as vertex attributes, if plot = TRUE.

If plot = TRUE, a network plot is displayed with nodes colored by module.

#' @details This function is a wrapper around several **igraph** community detection algorithms, including Louvain (cluster_louvain()), Walktrap, Infomap, Fast Greedy, and others. It simplifies their application and offers optional filtering, visualization via plot_Net(), and module subgraph extraction.

References

Csardi G, Nepusz T. The igraph software package for complex network research. InterJournal, Complex Systems. 2006;1695. https://igraph.org

Examples

```
## Not run:
library(igraph)
g <- sample_pa(100)
find_modules(g, method = "louvain", plot = TRUE)

## End(Not run)
```

greedy_seed_selection *Greedy Seed Node Selection to Maximize Diffusion Toward Target Nodes*

Description

This function implements a greedy algorithm to select a set of k seed nodes from a candidate list such that the resulting diffusion signal on a specified set of target nodes is maximized. It supports multiple diffusion models (Laplacian, Heat Kernel, Random Walk with Restart).

Usage

```
greedy_seed_selection(
  graph,
  target_nodes,
  candidate_nodes = NULL,
  method = c("laplacian", "heat", "rwr"),
  alpha = 0.7,
  t = 1,
  restart_prob = 0.3,
  k = 5,
  normalize = TRUE,
  plot = TRUE
)
```

Arguments

`graph`	An `igraph` object representing the network to analyze or a data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes.
`target_nodes`	A character vector of node names to prioritize for receiving the diffusion signal.
`candidate_nodes`	Optional character vector of eligible nodes to consider as seeds. If NULL (default), all non-target nodes are used.
`method`	Diffusion method to use. One of `"laplacian"`, `"heat"`, or `"rwr"`.
`alpha`	Scaling parameter for the Laplacian diffusion method (default = 0.7).
`t`	Time parameter for the Heat Kernel diffusion (default = 1).
`restart_prob`	Restart probability for the Random Walk with Restart (default = 0.3).
`k`	Number of seed nodes to select (default = 5).
`normalize`	Logical; whether to normalize the cumulative diffusion score over the target nodes (default = TRUE).
`plot`	Logical; whether to plot the progression of target diffusion score as seeds are selected (default = TRUE).

Details

This function uses a stepwise greedy heuristic. At each step, it evaluates the marginal gain in target score from adding each candidate node to the current seed set, and selects the one with the highest gain. This is repeated for `k` steps.

Note: The score of each candidate at every step is recomputed via a full diffusion run, making the function computationally intensive for large graphs or large `k`.

The target score can be interpreted as either the total or average diffusion signal received by the target nodes. Normalization helps scale results across networks of different sizes.

Value

A list with the following elements:

selected Character vector of selected seed node names.

final_target_score Final cumulative (or normalized) diffusion score over the target nodes.

scores_at_each_step Vector of target scores at each greedy selection step.

See Also

`network_diffusion`, `network_diffusion_with_pvalues`

Examples

```
## Not run:
g <- sample_gnp(50, 0.05, directed = F)
target <- c("1", "2", "3")
greedy_seed_selection(g, target_nodes = target, k = 10)

## End(Not run)
```

highlight_nodes *Highlight Nodes in a Network Plot*

Description

Highlights selected nodes in an `igraph` object by changing their label, fill color, and/or outline color. The function modifies node attributes and visualizes the result using `plot_Net()`.

Usage

```
highlight_nodes(
  graph,
  nodes,
  method = c("label", "fill", "outline"),
  label_color = "darkred",
  highlight_fill_color = "orange",
  highlight_frame_color = "darkblue",
  background_color = "gray",
  ...
)
```

Arguments

`graph`	An `igraph` object or a `data.frame` representing a symbolic edge list. If a `data.frame`, it should have at least two columns specifying source and target nodes.
`nodes`	A character vector of node names to highlight.
`method`	Character vector specifying how to highlight the nodes. One or more of: `"label"`, `"fill"`, `"outline"`.
`label_color`	Color for highlighted node labels. Used only if `"label"` is in `method`. Default is `"darkred"`.
`highlight_fill_color`	Fill color for highlighted nodes. Used only if `"fill"` is in `method`. Default is `"orange"`.
`highlight_frame_color`	Outline color for highlighted nodes. Used only if `"outline"` is in `method`. Default is `"darkblue"`.
`background_color`	Fill or frame color for non-highlighted nodes. Default is `"gray"`.
`...`	Additional arguments passed to `plot_Net()`.

Value

Invisibly returns the `igraph` object with updated attributes.

Examples

```
## Not run:
g <- igraph::make_ring(10)
igraph::V(g)$name <- letters[1:10]
highlight_nodes(g, nodes = c("a", "j"), method = c("label", "fill"))

## End(Not run)
```

`layout_horizontal_tree`
Horizontal Tree Layout for Graph Visualization

Description

Rotates a tree layout by -90 degrees to produce a horizontal orientation. This is particularly useful for hierarchical visualizations where a left-to-right structure is preferred over the default top-to-bottom tree layout.

Usage

```
layout_horizontal_tree(graph)
```

Arguments

`graph`	An `igraph` object representing the input graph.

Value

A numeric matrix with 2 columns representing x and y coordinates of each node in the layout. This matrix can be passed to `plot.igraph()` or other plotting functions.

Examples

```
## Not run:
g <- igraph::make_tree(10)
coords <- layout_horizontal_tree(g)
plot_Net(g, layout = coords)

## End(Not run)
```

`network_diffusion` *Perform Network Diffusion from Seed Nodes*

Description

Applies network diffusion techniques to propagate influence from a set of seed nodes across a graph. Supports Laplacian smoothing, heat diffusion, and random walk with restart (RWR).

Usage

```
network_diffusion(
  graph,
  seed_nodes,
  method = c("laplacian", "heat", "rwr"),
  alpha = 0.7,
  t = 1,
  restart_prob = 0.3,
  normalize = TRUE,
  precompute = NULL
)
```

Arguments

`graph`	An `igraph` object representing the network to analyze or a data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes. Must have named vertices.
`seed_nodes`	Character vector of seed node names (must match `V(graph)$name`).
`method`	Character. Diffusion method to use: • `"laplacian"`: Solves the linear system $(I + \alpha L)^{-1} f_0$, where L is the (normalized) graph Laplacian and α is a smoothing parameter. Internally, a sparse Cholesky decomposition is used for efficiency. • `"heat"`: Applies the heat diffusion model $e^{-tL} f_0$, where t controls diffusion time. A truncated Taylor expansion is used for approximation. • `"rwr"`: Random Walk with Restart. Iteratively solves $f = (1-r)Pf + rf_0$, where P is the transition matrix and r is the restart probability.
`alpha`	Damping factor for the Laplacian method. Default is `0.7`.
`t`	Time parameter for the heat diffusion method. Default is `1`.
`restart_prob`	Restart probability (usually between 0.3 and 0.7) for the RWR method. Default is `0.3`.
`normalize`	Logical. Whether to normalize the adjacency matrix (symmetric normalization for undirected graphs). Default is `TRUE`.
`precompute`	Optional list of precompute diffusion matrices (e.g., Laplacian, Cholesky factor, or transition matrix). Use `prepare_diffusion()` to generate this object and avoid redundant computations when calling this function repeatedly (e.g., in greedy optimization).

Details

This function allows flexible application of network diffusion strategies, useful in systems biology (e.g., gene prioritization, pathway propagation), network analysis, and disease gene discovery. The underlying matrix operations are based on well-established diffusion models from graph theory.

For `"rwr"` (random walk with restart), the algorithm iteratively propagates scores until convergence based on a row-normalized transition matrix. Recommended method for large networks.

For `"laplacian"` and `"heat"`, the graph Laplacian is computed from the (optionally normalized) adjacency matrix. For efficiency in iterative applications, precompute Laplacian and Cholesky decomposition using `prepare_diffusion()`.

Value

A data frame with two columns:

node Node name

score Diffusion score representing influence from the seed nodes

References

Köhler S, Bauer S, Horn D, Robinson PN. Walking the interactome for prioritization of candidate disease genes. *Am J Hum Genet*. 2008;82(4):949–958. doi:10.1016/j.ajhg.2008.02.013

Vanunu O, Magger O, Ruppin E, Shlomi T, Sharan R. Associating genes and protein complexes with disease via network propagation. *PLoS Comput Biol*. 2010;6(1):e1000641. doi:10.1371/journal.pcbi.1000641

Examples

```
## Not run:
g <- sample_gnp(100, 0.05, directed = F)
V(g)$name <- as.character(seq_len(vcount(g)))
seed_nodes <- sample(V(g)$name, 5)
network_diffusion(g, seed_npodes, method = "laplacian")

## End(Not run)
```

`network_diffusion_with_pvalues` *Perform Network Diffusion from Seed Nodes*

Description

Applies network diffusion techniques to propagate influence from a set of seed nodes across a graph. Supports Laplacian smoothing, heat diffusion, and random walk with restart (RWR).

Usage

```
network_diffusion_with_pvalues(
  graph,
  seed_nodes,
  method = c("laplacian", "heat", "rwr"),
  alpha = 0.7,
  t = 1,
  restart_prob = 0.3,
  normalize = TRUE,
  n_permutations = 1000,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`graph`	An `igraph` object representing the network to analyze or a data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes. Must have named vertices.
`seed_nodes`	Character vector of seed node names (must match `V(graph)$name`).
`method`	Character. Diffusion method to use: • `"laplacian"`: Solves the linear system $(I+\alpha L)^{-1}f_0$, where L is the (normalized) graph Laplacian and α is a smoothing parameter. Internally, a sparse Cholesky decomposition is used for efficiency. • `"heat"`: Applies the heat diffusion model $e^{-tL}f_0$, where t controls diffusion time. If available, a sparse approximation method is used to avoid dense matrix exponential. • `"rwr"`: Random Walk with Restart. Iteratively solves $f = (1-r)Pf + rf_0$, where P is the transition matrix and r is the restart probability.
`alpha`	Damping factor for the Laplacian method. Default is `0.7`.
`t`	Time parameter for the heat diffusion method. Default is `1`.
`restart_prob`	Restart probability (usually between 0.3 and 0.7) for the RWR method. Default is `0.3`.
`normalize`	Logical. Whether to normalize the adjacency matrix (symmetric normalization for undirected graphs). Default is `TRUE`.
`n_permutations`	Integer. Number of permutations to run for empirical p-value estimation (default `1000`).
`seed`	Optional integer for reproducible random number generation. If `NULL` (default), seed is not set.
`verbose`	Logical. If `TRUE` (default), displays a progress bar during permutations.

Details

This function allows flexible application of network diffusion strategies, useful in systems biology (e.g., gene prioritization, pathway propagation), network analysis, and disease gene discovery. The underlying matrix operations are based on well-established diffusion models from graph theory.

For `"rwr"` (random walk with restart), the algorithm iteratively propagates scores until convergence based on a row-normalized transition matrix. Recommended method for large networks.

For `"laplacian"` and `"heat"`, the graph Laplacian is computed from the (optionally normalized) adjacency matrix.

Value

A data frame with two columns:

node Node name

score Diffusion score representing influence from the seed nodes

References

Köhler S, Bauer S, Horn D, Robinson PN. Walking the interactome for prioritization of candidate disease genes. *Am J Hum Genet*. 2008;82(4):949–958. doi:10.1016/j.ajhg.2008.02.013

Vanunu O, Magger O, Ruppin E, Shlomi T, Sharan R. Associating genes and protein complexes with disease via network propagation. *PLoS Comput Biol*. 2010;6(1):e1000641. doi:10.1371/journal.pcbi.1000641

Examples

```
## Not run:
g <- sample_gnp(100, 0.05, directed = F)
V(g)$name <- as.character(seq_len(vcount(g)))
seed_nodes <- sample(V(g)$name, 5)
network_diffusion_with_pvalues(g, seed_npodes, method = "laplacian")

## End(Not run)
```

plot_CCDF *Plot Complementary Cumulative Degree Distribution (CCDF)*

Description

This function plots the complementary cumulative distribution function (CCDF) of node degrees in a network and optionally overlays power-law reference curves.

Usage

```
plot_CCDF(
  graph,
  keep_direction = TRUE,
  remove_singles = FALSE,
  show_PL = TRUE,
  PL_exponents = c(2, 3),
  colors = c("#000831", "#e41a1c", "darkgreen", "#9c52f2", "#b8b8ff"),
  label.size = 12
)
```

Arguments

`graph`	An `igraph` object representing the network to analyze or a data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes.

keep_direction	Logical. Only for directed graphs. If TRUE, CCDF curves are drawn for 'in'-dgree, 'out'-degree, and 'all'-degree distributions. FALSE to ignore directionality.
remove_singles	Logical. If TRUE, nodes with degree 0 are removed from the graph before computing the CCDF. Default is FALSE.
show_PL	Logical. If TRUE, overlays theoretical power-law reference lines of the form $P(K > k) \sim k^{-\gamma}$. Default is TRUE.
PL_exponents	Numeric vector. The γ exponents for the power-law curves. Default is c(2, 3).
colors	Optional character vector. Custom colors for the graph curve and power-law lines. If NULL, default colors are used.
label.size	Numeric. Font size for axis labels and theme. Passed to theme_minimal.

Value

A ggplot2 object showing the CCDF of node degrees on a log-log scale.

Examples

```
## Not run:
library(igraph)
g <- sample_pa(1000)
plot_CCDF(g, remove_singles = TRUE)

## End(Not run)
```

plot_Net — *Plot an igraph network with customizable node sizes and edge widths*

Description

This function plots an igraph network graph with options to customize node size based on vertex degree or a numeric vertex attribute, set node and edge colors, label nodes, and adjust edge widths (optionally mapped to edge betweenness centrality).

Usage

```
plot_Net(
  graph,
  label = FALSE,
  color = "#006d77",
  color_ramp = c("blue", "white", "red"),
  NA_color = "gray",
  frame.color = NULL,
  node.size.factor = 1,
  node.degree.map = TRUE,
  edge.color = "#999999",
  edge.width.factor = 1,
  edge.bw.map = TRUE,
  label.color = "#e29578",
  label.size = 1,
  layout = NULL,
  ...
)
```

Arguments

graph	An igraph object representing the network to plot or a data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes.
label	Logical, or character vector. If FALSE, no node labels are shown. If TRUE, node labels are set to vertex names or existing labels. If a character vector of length equal to the number of vertices, used as custom node labels.
color	Character. Color for node fill, or existing vertex attribute name to map node fill color. Default is '#006d77'.
color_ramp	A character vector of colors (e.g., c("blue", "white", "red")) used to create a continuous color ramp for mapping numeric node attributes to colors. Ignored if a fixed color is provided or if node color is categorical. Passed to colorRampPalette to interpolate a gradient of colors.
NA_color	Node color for NAs in node color mapping.
frame.color	Character. Color for node frame, or existing vertex attribute name to map node frame color. Default is NULL.
node.size.factor	Numeric or character. If numeric, acts as a constant scaling factor for node size. If character, it should be the name of a numeric vertex attribute used to size nodes.
node.degree.map	Logical. If TRUE and node.size.factor is numeric, node size is mapped to vertex degree multiplied by node.size.factor. Ignored if node.size.factor is character.
edge.color	Character. Color of edges. Default is '#999999'.
edge.width.factor	Numeric. Factor to scale edge widths. Default is 1.
edge.bw.map	Logical. If TRUE, edge widths are mapped to the log-transformed edge betweenness centrality. Otherwise, edges have uniform width.
label.color	Character. Color of node labels. Default is '#e29578'.
label.size	Numeric. Character expansion factor for label size. Default is 1.
layout	Optional numeric matrix specifying vertex coordinates for layout.
...	Additional parameters passed to plot.igraph.

Value

Invisibly returns NULL. The function produces a plot.

Examples

```
## Not run:
library(igraph)
g <- random.graph.game(100, 0.02)
plot_Net(g, label = TRUE, node.size.factor = 2)
```

```
## End(Not run)
```

`robustness_analysis` *Network Robustness Analysis via Node Removal Simulation*

Description

Simulates the removal of nodes from a network using various strategies and evaluates how the structure degrades using selected robustness metrics. Useful for assessing the vulnerability or resilience of a graph.

Usage

```
robustness_analysis(
  graph,
  removal_strategy = c("random", "degree", "betweenness"),
  steps = 50,
  metrics = c("lcc_size", "efficiency", "n_components"),
  n_reps = 50,
  plot = TRUE,
  seed = NULL
)
```

Arguments

`graph`	An `igraph` object representing the network to plot or a data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes. Must be undirected; directed graphs will be converted.
`removal_strategy`	Character. Strategy used for node removal. Options are: `"random"`, `"degree"`, `"betweenness"`, or the name of a numeric vertex attribute. Custom attributes are interpreted as priority scores (higher = removed first).
`steps`	Integer. Number of removal steps (default: 50).
`metrics`	Character vector. Structural metrics to compute at each step. Options include: `"lcc_size"`, `"efficiency"`, and `"n_components"`.
`n_reps`	Integer. Number of simulation repetitions (only relevant if `removal_strategy = "random"`).
`plot`	Logical. If `TRUE`, a robustness plot is generated.
`seed`	Integer or NULL. Random seed for reproducibility.

Details

This function builds on classic approaches in network science for evaluating structural robustness (e.g., Albert, Jeong, & Barabási, Nature 2000) by simulating progressive node removal and quantifying the degradation of key topological features.

For deterministic strategies (`"degree"`, `"betweenness"`, or custom attributes), nodes are removed in a fixed priority order. For the `"random"` strategy, the process is repeated `n_reps` times, and the results are aggregated.

#' @references Albert R, Jeong H, Barabási AL. Error and attack tolerance of complex networks. Nature. 2000;406(6794):378–382. doi:10.1038/35019019

The function uses:

- **Largest Connected Component (LCC)**: Size of the largest remaining component.
- **Global Efficiency**: Average inverse shortest path length among all pairs.
- **Number of Components**: Total number of disconnected components.

Additionally, Area Under the Curve (AUC) is calculated for each metric, providing a scalar summary of robustness. A higher AUC indicates greater resilience (i.e., slower degradation).

The implementation is inspired by principles described in: Albert R, Jeong H, Barabási AL. *Error and attack tolerance of complex networks*. Nature. 2000;406:378–382. (doi:10.1038/35019019)

The function uses:

- Size of the largest connected component (`lcc_size`)
- Global efficiency (average inverse shortest path length)
- Number of components (`n_components`)

to evaluate how robust the network remains during progressive node failure.

Value

A list with:

`all_results` A data frame with simulation results across all steps and repetitions.

`summary` A summarized data frame (mean and SD) if `n_reps > 1`, otherwise raw results.

`auc` Named list of AUC (area under the curve) values for each selected metric.

If `plot = TRUE`, a ggplot object is generated showing the evolution of selected metrics as nodes are progressively removed.

Examples

```
## Not run:
g <- igraph::sample_pa(100)
robustness_analysis(g, removal_strategy = "degree", metrics = c("lcc_size", "n_components"))

## End(Not run)
```

`summarize_graph_metrics` *Summarize Topological Properties of a Graph*

Description

Computes a comprehensive set of global topological metrics for an input graph, including basic structure, connectivity, spectral properties, and complexity. Supports both `igraph` objects and data frames representing edge lists.

Usage

```
summarize_graph_metrics(graph)
```

Arguments

`graph`	An `igraph` object or a data frame with columns `from` and `to` representing an edge list.

Details

Metrics computed:

- Number of nodes and edges
- Directed TRUE/FALSE
- Graph density
- Diameter and average path length of the largest connected component
- Clustering coefficient (transitivity)
- Degree assortativity
- Average degree and betweenness centrality
- Number of connected components and size of the largest connected component
- Number of single nodes
- Algebraic connectivity (second-smallest Laplacian eigenvalue)
- Degree entropy (Shannon entropy of the degree distribution)
- Gini coefficient of node degrees
- Modularity of the community structure (via Louvain algorithm)

Value

A tibble with one row and multiple columns, each representing a graph-level metric.

References

- Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press.
- Estrada, E. (2012). *The Structure of Complex Networks: Theory and Applications*. Oxford University Press.
- Latora, V., Nicosia, V., & Russo, G. (2017). *Complex Networks: Principles, Methods and Applications*. Cambridge University Press.
- Louvain modularity method: Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). *Fast unfolding of communities in large networks*. J. Stat. Mech., 2008(10), P10008.

Examples

```
## Not run:
g <- igraph::sample_gnp(200, 0.05, directed = F)
summarize_graph_metrics(g)

## End(Not run)
```

Index