

Package: ig.degree.betweenness (via r-universe)

October 18, 2026

Type Package

Title ``Node+Edge Betweenness Community Detection Algorithm for 'igraph' Objects"

Version 0.2.0

Description Implements the ``Node + Edge Betweenness" community detection algorithm for network analysis using 'igraph' objects. This algorithm combines node degree and betweenness centrality measures to identify communities within networks, with a gradient evident in social partitioning. The package provides functions for implementation of the algorithm, visualization, and analysis of the resulting community structure along with the original dataset used in the publication. Methods are based on results from Smith, Pittman and Xu (2026) <[doi:10.1002/cjs.70060](https://doi.org/10.1002/cjs.70060)>.

License MIT + file LICENSE

URL <https://github.com/benyamindsmith/ig.degree.betweenness>,
<https://benyamindsmith.github.io/ig.degree.betweenness/>

BugReports <https://github.com/benyamindsmith/ig.degree.betweenness/issues>

Depends R (>= 4.1.0)

Imports BBmisc, dplyr, ggplot2, igraph, igraphdata, lifecycle, qgraph,
rlist, stats, tibble, tidyr

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs

cmake libglpk-dev make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev

Repository <https://benyamindsmith.r-universe.dev>

Date/Publication 2026-06-17 16:26:32 UTC

RemoteUrl <https://github.com/benyamindsmith/ig.degree.betweenness>
RemoteRef HEAD
RemoteSha 96c8c868c9cfba1ac436b9aa1f327424096f7e4d

Contents

cluster_degree_betweenness	2
degree_balance	3
lcd_graphs	4
oncology_network	6
plot_node_degrees	7
plot_simplified_edgeplot	8
prep_unlabeled_graph	9
Index	11

cluster_degree_betweenness	<i>Community structure detection based on node degree centrality and edge betweenness</i>
----------------------------	---

Description

Referred to as the "Smith-Pittman" algorithm in Smith et al (2024). This algorithm detects communities by calculating the degree centrality measures of nodes and edge betweenness.

Usage

```
cluster_degree_betweenness(graph)
```

Arguments

graph	The graph to analyze
-------	----------------------

Details

This can be thought of as an alternative version of `igraph::cluster_edge_betweenness()`. The function iteratively removes edges based on their betweenness centrality and the degree of their adjacent nodes. At each iteration, it identifies the edge with the highest betweenness centrality among those connected to nodes with the highest degree. It then removes that edge and recalculates the modularity of the resulting graph. The process continues until all edges have been assessed or until no further subgraph can be created with the optimal number of communities being chosen based on maximization of modularity.

Value

An `igraph` "communities" object with detected communities via the Smith-Pittman algorithm.

References

Smith et al (2024) "Centrality in Collaboration: A Novel Algorithm for Social Partitioning Gradients in Community Detection for Multiple Oncology Clinical Trial Enrollments", [doi:10.48550/arXiv.2411.01394](https://doi.org/10.48550/arXiv.2411.01394)

Examples

```
library(igraphdata)
data("karate")
ndb <- cluster_degree_betweenness(karate)
plot(
  ndb,
  karate,
  main= "Degree-Betweenness Clustering"
)

ndb

data("UKfaculty")
# Making graph undirected so it looks nicer when its plotted
uk_faculty <- UKfaculty |>
  igraph::as.undirected()

ndb <- cluster_degree_betweenness(uk_faculty)

plot(
  ndb,
  uk_faculty,
  main= "Smith-Pittman Clustering for UK Faculty"
)
```

degree_balance	Analyze Degree Balance of a Graph
----------------	-----------------------------------

Description

Computes and summarizes vertex degree distributions for an igraph object. For directed graphs, the function reports in-degree and out-degree statistics, including their correlation and mean values. For undirected graphs, only overall degree statistics are reported.

Usage

```
degree_balance(g)
```

Arguments

g An igraph graph object.

Details

If the graph is directed (see `igraph::is_directed()`), the function:

- Computes in-degree and out-degree for each vertex.
- Prints summary statistics for both distributions.
- Reports the correlation between in-degree and out-degree.
- Reports mean in-degree and mean out-degree.

If the graph is undirected, the function:

- Computes the degree for each vertex.
- Prints summary statistics of the degree distribution.

Value

A list containing:

- `out_degree` Numeric vector of out-degrees (directed graphs only).
- `in_degree` Numeric vector of in-degrees (directed graphs only).
- `degree` Numeric vector of degrees (undirected graphs only).

Examples

```
library(igraph)
library(ig.degree.betweenness)

# Directed graph example
g_directed <- make_ring(10, directed = TRUE)
degree_balance(g_directed)

# Undirected graph example
g_undirected <- make_ring(10)
degree_balance(g_undirected)
```

lcd_graphs

Generate Linearized Chord Diagram (LCD) Graphs

Description

[Experimental]

A family of functions to generate networks based on the Linearized Chord Diagram (LCD) model using preferential attachment. The suite includes the standard LCD model as well as variants that introduce bidirectional edges, alternating direction attachment, and mixed degree preferences.

Usage

```
lcd_graph(n, m = 1, directed = TRUE)

lcd_graph_bidirectional(n, m = 1, directed = TRUE, bidirectional_prob = 0.5)

lcd_graph_alternating(n, m = 1, directed = TRUE)

lcd_graph_mixed(n, m = 1, directed = TRUE, out_weight = 0.5)
```

Arguments

n	Integer. The total number of vertices (nodes) in the generated graph.
m	Integer. The number of edges each new node adds during its time step. Defaults to 1.
directed	Logical. Whether the generated graph should be directed. Defaults to TRUE.
bidirectional_prob	Numeric. The probability (between 0 and 1) of adding a reverse edge (target -> source) when generating bidirectional graphs. Used only in <code>lcd_graph_bidirectional</code> . Defaults to 0.5.
out_weight	Numeric. The probability weight (between 0 and 1) given to the out-degree pool versus the in-degree pool. Used only in <code>lcd_graph_mixed</code> . Defaults to 0.5.

Details

The standard `lcd_graph` function builds a network step-by-step. At each time step t from 1 to n , a new node is added and generates m edges. The target of each edge is chosen preferentially based on the degree of the existing nodes.

The package also provides three variations:

- `lcd_graph_bidirectional`: Adds a primary edge and, with a specified probability, simultaneously adds a reverse edge.
- `lcd_graph_alternating`: Alternates between using out-degree and in-degree pools for preferential attachment targets.
- `lcd_graph_mixed`: Chooses between out-degree and in-degree pools for attachment based on a weighted probability.

Value

An igraph object representing the generated LCD graph.

References

Barabási, A.-L. (2016). *Network Science*. Cambridge University Press. Chapter 5: The Barabási-Albert Model. <https://networksciencebook.com/chapter/5>

Examples

```
# Generate a standard directed LCD graph with 100 nodes and 2 edges per step
g1 <- lcd_graph(n = 100, m = 2)

# Generate a graph where reverse edges appear 30% of the time
g2 <- lcd_graph_bidirectional(n = 100, m = 1, bidirectional_prob = 0.3)

# Generate a graph alternating between in-degree and out-degree attachment
g3 <- lcd_graph_alternating(n = 100, m = 3)

# Generate a mixed graph favoring out-degree attachment 70% of the time
g4 <- lcd_graph_mixed(n = 100, m = 2, out_weight = 0.7)
```

oncology_network

Oncology Clinical Trial Referral Network

Description

A simulated oncology clinical trial referral network from a major research hospital. For the purpose of identifying collaboration networks between oncologists, this dataset only includes referrals of patients who were enrolled in more than one clinical trial. This includes 389 patients enrolled in 288 clinical trials.

Usage

```
oncology_network
```

Format

A igraph object (e.g. representing the oncology clinical trial referral network. The structure includes:

nodes Oncologists or clinical trials (depending on network structure).

edges Referral links between nodes, based on shared patient enrollment across trials.

Details

Clinical trials are categorized by intervention type, including targeted therapies (prefixed with "T:") and immunotherapies (prefixed with "I:"). There are 16 distinct intervention types (nodes) and 470 patient referrals (edges) in the network.

Source

Simulated data based on oncology clinical trial enrollment patterns.

plot_node_degrees	<i>Visualize Node Degree Distribution in a Network Graph</i>
-------------------	--

Description

Generates a horizontal bar-style plot of node degrees for an igraph network. For undirected graphs, it shows each node's total degree. For directed graphs, it displays in-degrees (as negative bars) alongside out-degrees.

Usage

```
plot_node_degrees(graph)
```

Arguments

`graph` An igraph object. Can be either directed or undirected.

Details

This function computes:

Total degree Number of edges incident on each node (for undirected graphs).

In-degree Number of incoming edges per node (for directed graphs).

Out-degree Number of outgoing edges per node (for directed graphs).

For directed graphs, in-degrees are negated so that bars extend leftward, providing an immediate visual comparison to out-degrees.

Internally, it uses:

- `igraph::degree()` to compute degrees,
- `dplyr` and `tidyr` for reshaping the data,
- `ggplot2` for plotting.

Value

A ggplot object:

- **Undirected graphs:** A bar for each node showing its total degree.
- **Directed graphs:** Split bars per node with negative values for in-degree (pointing left) and positive values for out-degree (pointing right).

Customization

You can modify the returned ggplot with additional layers, themes, or labels. For example, to add a title or change colors:

```
plot_node_degrees(g) +  
  ggtitle("Degree Distribution") +  
  scale_fill_manual(values = c("in_degree" = "steelblue", "out_degree" = "salmon"))
```

Examples

```
library(ig.degree.betweenness)
library(igraphdata)
data("karate")
data("oncology_network")
plot_node_degrees(oncology_network)
plot_node_degrees(karate)
```

```
plot_simplified_edgeplot
```

Plot Simplified Edgeplot

Description

This function generates a simplified edge plot of an igraph object, optionally highlighting communities if provided.

Usage

```
plot_simplified_edgeplot(graph, communities = NULL, edge.arrow.size = 0.2, ...)
```

Arguments

graph	igraph object
communities	optional; A communities object
edge.arrow.size	edge.arrow size arg. See ?igraph::plot.igraph for more details
...	other arguments to be passed to the plot() function

Details

This function is ideally for networks with a low number of nodes having varying numbers of connection and self loops. See the example for a better visual understanding.

Value

No return value, called for side effects.

Examples

```
# Load the igraph package
library(igraph)
library(ig.degree.betweenness)
# Set parameters
num_nodes <- 15    # Number of nodes (adjust as needed)
initial_edges <- 1  # Starting edges for preferential attachment
```

```

# Create a directed, scale-free network using the Barabási-Albert model
g <- sample_pa(n = num_nodes, m = initial_edges, directed = TRUE)

# Introduce additional edges to high-degree nodes to accentuate popularity differences
num_extra_edges <- 350 # Additional edges to create more popular nodes
set.seed(123)          # For reproducibility

for (i in 1:num_extra_edges) {
  # Sample nodes with probability proportional to their degree (to reinforce popularity)
  from <- sample(V(g), 1, prob = degree(g, mode = "in") + 1) # +1 to avoid zero probabilities
  to <- sample(V(g), 1)

  # Ensure we don't add the same edge repeatedly unless intended, allowing self-loops
  g <- add_edges(g, c(from, to))
}

# Add self-loops to a subset of nodes
num_self_loops <- 5
for (i in 1:num_self_loops) {
  node <- sample(V(g), 1)
  g <- add_edges(g, c(node, node))
}

ig.degree.betweenness::plot_simplified_edgeplot(g, main="Simulated Data")

```

prep_unlabeled_graph	<i>Prepared Unlabeled Graph to work with Degree-Betweenness Algorithm</i>
----------------------	---

Description

[Deprecated] Presently, `cluster_degree_betweenness()` function only works with labeled graphs. `prep_unlabeled_graph()` is a utility function that gives an unlabeled graph labels which are string values of their vertices.

Usage

```
prep_unlabeled_graph(graph)
```

Arguments

graph an unlabeled graph.

Value

An "igraph" object with named vertices.

See Also

[cluster_degree_betweenness\(\)](#) which this function aids.

Examples

```
library(igraph)
library(igraphdata)
library(ig.degree.betweenness)
data("UKfaculty")
# Making graph undirected so it looks nicer when its plotted
uk_faculty <- prep_unlabeled_graph(UKfaculty) |>
  as.undirected()

ndb <- cluster_degree_betweenness(uk_faculty)

plot(
  ndb,
  uk_faculty,
  main= "Node Degree Clustering"
)

ndb
```

Index

`cluster_degree_betweenness`, [2](#)
`cluster_degree_betweenness()`, [10](#)

`degree_balance`, [3](#)

`lcd_graph` (`lcd_graphs`), [4](#)
`lcd_graph_alternating` (`lcd_graphs`), [4](#)
`lcd_graph_bidirectional` (`lcd_graphs`), [4](#)
`lcd_graph_mixed` (`lcd_graphs`), [4](#)
`lcd_graphs`, [4](#)

`oncology_network`, [6](#)

`plot_node_degrees`, [7](#)
`plot_simplified_edgeplot`, [8](#)
`prep_unlabeled_graph`, [9](#)