Package ‘influential’

February 17, 2020

Type Package

Title Identification of the Most Influential Nodes (Hubs)

Version 0.1.0

Author Adrian (Abbas) Salavaty

Maintainer Adrian Salavaty <abbas.salavaty@gmail.com>

Description Contains functions for reconstruction of networks from adjacency matrices and data frames, analysis of the topology of the network and calculation of centrality measures, and identification of the most influential nodes (network hubs). Also, some functions have been provided for the assessment of dependence and correlation of two network centrality measures as well as the conditional probability of deviation from their corresponding means in opposite direction.


Imports igraph

Suggests Hmisc (>= 4.3-0), mgcv (>= 1.8-31), nortest (>= 1.0-4), NNS (>= 0.4.7.1), parallel, knitr, rmarkdown

Depends R (>= 2.10)

URL http://github.com/asalavaty/influential

BugReports http://github.com/asalavaty/influential/issues

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2.9000

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-02-17 14:50:02 UTC
Vertex betweenness centrality

Description

This function and all of its descriptions have been obtained from the igraph package.

Usage

betweenness(
  graph,
  v = V(graph),
  directed = TRUE,
  weights = NULL,
  nobigint = TRUE,
  normalized = FALSE
)

Arguments

- graph: The graph to analyze (an igraph graph).
- v: The vertices for which the vertex betweenness will be calculated.
- directed: Logical, whether directed paths should be considered while determining the shortest paths.
- weights: Optional positive weight vector for calculating weighted betweenness. If the graph has a weight edge attribute, then this is used by default. Weights are used to calculate weighted shortest paths, so they are interpreted as distances.
centrality.measures

bezbigint Logical scalar, whether to use big integers during the calculation. This is only required for lattice-like graphs that have very many shortest paths between a pair of vertices. If TRUE (the default), then big integers are not used.

normalized Logical scalar, whether to normalize the betweenness scores. If TRUE, then the results are normalized.

Value

A numeric vector with the betweenness score for each vertex in v.

See Also

betweenness for a complete description on this function

Other centrality functions: degree(), neighborhood.connectivity()

Examples

MyData <- coexpression.data
My_graph <- graph_from_data_frame(MyData)
GraphVertices <- V(My_graph)
My_graph_betweenness <- betweenness(My_graph, v = GraphVertices, directed = FALSE, normalized = FALSE)

centrality.measures

Centrality measures dataset

Description

The centrality measures of a co-expression network of lncRNAs and mRNAs in lung adenocarcinoma

Usage

centrality.measures

Format

A data frame with 794 rows and 4 variables:

<table>
<thead>
<tr>
<th>BetweennessCentrality</th>
<th>Betweenness Centrality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>Degree Centrality</td>
</tr>
<tr>
<td>name</td>
<td>Node (gene) name</td>
</tr>
<tr>
<td>NeighborhoodConnectivity</td>
<td>Neighborhood Connectivity ...</td>
</tr>
</tbody>
</table>

Source

coexpression.adjacency

*Adjacency matrix*

**Description**

The adjacency matrix of a co-expression network of IncRNAs and mRNAs in lung adenocarcinoma that was generated using igraph functions

**Usage**

coexpression.adjacency

**Format**

A data frame with 794 rows and 794 variables:

- **IncRNA** IncRNA symbol
- **IncRNA** IncRNA symbol ...

**Source**


---

coexpression.data

*Co-expression dataset*

**Description**

A co-expression dataset of IncRNAs and mRNAs in lung adenocarcinoma

**Usage**

coexpression.data

**Format**

A data frame with 2410 rows and 2 variables:

- **IncRNA** IncRNA symbol
- **Coexpressed.Gene** Co-expressed gene symbol ...

**Source**

cond.prob.analysis

Conditional probability of deviation from means

Description

This function calculates the conditional probability of deviation of two centrality measures (or any two other continuous variables) from their corresponding means in opposite directions.

Usage

cond.prob.analysis(data, nodes.colname, Desired.colname, Condition.colname)

Arguments

data A data frame containing the values of two continuous variables and the name of observations (nodes).
nodes.colname The character format (quoted) name of the column containing the name of observations (nodes).
Desired.colname The character format (quoted) name of the column containing the values of the desired variable.
Condition.colname The character format (quoted) name of the column containing the values of the condition variable.

Value

A list of two objects including the conditional probability of deviation of two centrality measures (or any two other continuous variables) from their corresponding means in opposite directions based on both the entire network and the split-half random sample of network nodes.

See Also

Other centrality association assessment functions: double.cent.assess.noRegression(), double.cent.assess()

Examples

MyData <- centrality.measures
My.conditional.prob <- cond.prob.analysis(data = MyData, 
nodes.colname = "name", 
Desired.colname = "BetweennessCentrality", 
Condition.colname = "NeighborhoodConnectivity")
**Degree of the vertices**

Description

This function and all of its descriptions have been obtained from the igraph package.

Usage

```
degree(
  graph,
  v = V(graph),
  mode = c("all", "out", "in", "total"),
  loops = TRUE,
  normalized = FALSE
)
```

Arguments

- `graph` The graph to analyze (an igraph graph).
- `v` The ids of vertices of which the degree will be calculated.
- `mode` Character string, "out" for out-degree, "in" for in-degree or "total" for the sum of the two. For undirected graphs this argument is ignored. "all" is a synonym of "total".
- `loops` Logical; whether the loop edges are also counted. If the graph has a weight edge attribute, then this is used by default. Weights are used to calculate weighted shortest paths, so they are interpreted as distances.
- `normalized` Logical scalar, whether to normalize the degree. If TRUE then the result is divided by n-1, where n is the number of vertices in the graph.

Value

A numeric vector of the same length as argument `v`.

See Also

- `degree` for a complete description on this function
- Other centrality functions: `betweenness()`, `neighborhood.connectivity()`

Examples

```
MyData <- coexpression.data
My_graph <- graph_from_data_frame(MyData)
GraphVertices <- V(My_graph)
My_graph_degree <- degree(My_graph, v = GraphVertices, normalized = FALSE)
```
double.cent.assess

Assessment of innate features and associations of two network centrality measures (dependent and independent)

Description

This function assesses innate features and the association of two centrality measures (or any two other continuous variables) from the aspect of distribution mode, dependence, linearity, monotonicity, partial-moments based correlation, and conditional probability of deviating from corresponding means in opposite direction. This function assumes one variable as dependent and the other as independent for regression analyses. The non-linear nature of the association of two centrality measures is evaluated based on generalized additive models (GAM). The monotonicity of the association is evaluated based on comparing the squared coefficient of Spearman correlation and R-squared of rank regression analysis. Also, the correlation between two variables is assessed via non-linear non-parametric statistics (NNS). For the conditional probability assessment, the independent variable is considered as the condition variable.

Usage

double.cent.assess(
  data,
  nodes.colname,
  dependent.colname,
  independent.colname,
  plot = FALSE
)

Arguments

data A data frame containing the values of two continuous variables and the name of observations (nodes).
nodes.colname The character format (quoted) name of the column containing the name of observations (nodes).
dependent.colname The character format (quoted) name of the column containing the values of the dependent variable.
independent.colname The character format (quoted) name of the column containing the values of the independent variable.
plot logical; FALSE (default) Plots quadrant means of NNS correlation analysis.

Value

A list of 11 objects including: Summary of the basic statistics of two centrality measures (or any two other continuous variables) The results of normality assessment of two variable (p-value > 0.05 imply that the variable is normally distributed). Description of the normality assessment of the
dependent variable. Description of the normality assessment of the independent variable. Results of the generalized additive modeling (GAM) of the data. The association type based on simultaneous consideration of normality assessment, GAM Computation with smoothness estimation, Spearman correlation, and ranked regression analysis of splines. The Hoeffding’s D Statistic of dependence (ranging from -0.5 to 1). Description of the dependence significance. Correlation between variables based on the NNS method. The last two objects are the conditional probability of deviation of two centrality measures from their corresponding means in opposite directions based on both the entire network and the split-half random sample of network nodes.

See Also

ad.test for Anderson-Darling test for normality, gam for Generalized additive models with integrated smoothness estimation, lm for Fitting Linear Models, hoeffd for Matrix of Hoeffding’s D Statistics, and NNS.dep for NNS Dependence

Other centrality association assessment functions: cond.prob.analysis(), double.cent.assess.noRegression()

Examples

MyData <- centrality.measures
My.metrics.assessment <- double.cent.assess(data = MyData,
  nodes.colname = "name",
  dependent.colname = "BetweennessCentrality",
  independent.colname = "NeighborhoodConnectivity")

double.cent.assess.noRegression
  Assessment of innate features and associations of two network centrality measures

Description

This function assesses innate features and the association of two centrality measures (or any two other continuous variables) from the aspect of distribution mode, dependence, linearity, partial-moments based correlation, and conditional probability of deviating from corresponding means in opposite direction (centrality2 is used as the condition variable). This function doesn’t consider which variable is dependent and which one is independent and no regression analysis is done. Also, the correlation between two variables is assessed via non-linear non-parametric statistics (NNS). For the conditional probability assessment, the centrality2 variable is considered as the condition variable.

Usage

double.cent.assess.noRegression(  
data,  
  nodes.colname,  
  centrality1.colname,  
  centrality2.colname  )
Arguments

- **data**: A data frame containing the values of two continuous variables and the name of observations (nodes).
- **nodes.colname**: The character format (quoted) name of the column containing the name of observations (nodes).
- **centrality1.colname**: The character format (quoted) name of the column containing the values of the Centrality_1 variable.
- **centrality2.colname**: The character format (quoted) name of the column containing the values of the Centrality_2 variable.

Value

A list of nine objects including:
- Summary of the basic statistics of two centrality measures (or any two other continuous variables)
- The results of normality assessment of two variable (p-value > 0.05 imply that the variable is normally distributed).
- Description of the normality assessment of the centrality1 (first variable).
- Description of the normality assessment of the centrality2 (second variable).
- The Hoeffding’s D Statistic of dependence (ranging from -0.5 to 1).
- Description of the dependence significance.
- Correlation between variables based on the NNS method.
- The last two objects are the conditional probability of deviation of two centrality measures from their corresponding means in opposite directions based on both the entire network and the split-half random sample of network nodes.

See Also

- `ad.test` for Anderson-Darling test for normality, `hoeffd` for Matrix of Hoeffding’s D Statistics, and `NNS.dep` for NNS Dependence
- Other centrality association assessment functions: `cond.prob.analysis()`, `double.cent.assess()`

Examples

```r
MyData <- centrality.measures
My.metrics.assessment <- double.cent.assess.noRegression(data = MyData,
 nodes.colname = "name",
 centrality1.colname = "BetweennessCentrality",
 centrality2.colname = "NeighborhoodConnectivity")
```

---

**graph_from_adjacency_matrix**

*Creating igraph graphs from adjacency matrices*

**Description**

This function and all of its descriptions have been obtained from the igraph package. For a complete description if the function and its arguments try this: `?igraph::graph_from_adjacency_matrix`
Usage

```r
graph_from_adjacency_matrix(
  adjmatrix,
  mode = c("directed", "undirected", "max", "min", "upper", "lower", "plus"),
  weighted = NULL,
  diag = TRUE,
  add.colnames = NULL,
  add.rownames = NA
)
```

Arguments

- `adjmatrix`: A square adjacency matrix. From igraph version 0.5.1 this can be a sparse matrix created with the Matrix package.
- `mode`: Character scalar, specifies how igraph should interpret the supplied matrix. See also the `weighted` argument, the interpretation depends on that too. Possible values are: directed, undirected, upper, lower, max, min, plus.
- `weighted`: This argument specifies whether to create a weighted graph from an adjacency matrix. If it is NULL then an unweighted graph is created and the elements of the adjacency matrix gives the number of edges between the vertices. If it is a character constant then for every non-zero matrix entry an edge is created and the value of the entry is added as an edge attribute named by the `weighted` argument. If it is TRUE then a weighted graph is created and the name of the edge attribute will be `"weight"`.
- `diag`: Logical scalar, whether to include the diagonal of the matrix in the calculation. If this is FALSE then the diagonal is zeroed out first.
- `add.colnames`: Character scalar, whether to add the column names as vertex attributes. If it is ‘NULL’ (the default) then, if present, column names are added as vertex attribute ‘name’. If ‘NA’ then they will not be added. If a character constant, then it gives the name of the vertex attribute to add.
- `add.rownames`: Character scalar, whether to add the row names as vertex attributes. Possible values the same as the previous argument. By default row names are not added. If ‘add.rownames’ and ‘add.colnames’ specify the same vertex attribute, then the former is ignored.

Value

An igraph graph object.

See Also

- `graph_from_adjacency_matrix` for a complete description on this function
- Other network reconstruction functions: `graph_from_data_frame`
graph_from_data_frame

Examples

```r
MyData <- coexpression.adjacency
My_graph <- graph_from_adjacency_matrix(MyData)
```

desc

Description

This function and all of its descriptions have been obtained from the igraph package. For a complete description if the function and its arguments try this: ?igraph::graph_from_data_frame

Usage

```r
graph_from_data_frame(d, directed = TRUE, vertices = NULL)
```

Arguments

d A data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes. Since version 0.7 this argument is coerced to a data frame with as.data.frame.
directed Logical scalar, whether or not to create a directed graph.
vertices A data frame with vertex metadata, or NULL. Since version 0.7 of igraph this argument is coerced to a data frame with as.data.frame, if not NULL.

Value

An igraph graph object.

See Also

- `graph_from_adjacency_matrix` for a complete description on this function
- Other network_reconstruction functions: `graph_from_adjacency_matrix()`

Examples

```r
MyData <- coexpression.data
My_graph <- graph_from_data_frame(d=MyData)
```
### ihs

**Integrated hubness score (IHS)**

#### Description

This function calculates the IHS of the desired nodes. This function is not dependent to other packages and the required centrality measures, namely degree centrality, betweenness centrality and neighborhood connectivity could have been calculated by any means beforehand.

#### Usage

```r
ihs(DC, BC, NC)
```

#### Arguments

- **DC**
  - A vector containing the values of degree centrality of the desired vertices.
- **BC**
  - A vector containing the values of betweenness centrality of the desired vertices.
- **NC**
  - A vector containing the values of neighborhood connectivity of the desired vertices.

#### Value

A numeric vector with the IHS score based on the provided centrality measures.

#### Examples

```r
MyData <- centrality.measures
My.vertices.IHS <- ihs(DC = centrality.measures$Degree,
                      BC = centrality.measures$BetweennessCentrality,
                      NC = centrality.measures$NeighborhoodConnectivity)
```

### neighborhood.connectivity

**Neighborhood connectivity**

#### Description

This function calculates the neighborhood connectivity of input vertices and works with both directed and undirected networks.

#### Usage

```r
neighborhood.connectivity(graph, vertices, mode = "all")
```
Arguments

- **graph**: A graph (network) of the igraph class.
- **vertices**: A vector of desired vertices, which could be obtained by the V function.
- **mode**: The mode of neighborhood connectivity depending on the directedness of the graph. If the graph is undirected, the mode "all" should be specified. Otherwise, for the calculation of neighborhood connectivity based on incoming connections select "in" and for the outgoing connections select "out". Also, if all of the connections are desired, specify the "all" mode. Default mode is set to "all".

Value

A one column data frame with vertex names in vertices as row names and neighborhood connectivity score of each vertex in the column.

See Also

Other centrality functions: `betweenness()`, `degree()`

Examples

```r
MyData <- coexpression.data
My_graph <- graph_from_data_frame(MyData)
GraphVertices <- V(My_graph)
neighborhood.co <- neighborhood.connectivity(graph = My_graph,
                                             vertices = GraphVertices,
                                             mode = "all")
```

---

**V**

*Vertices of an igraph graph*

Description

This function and all of its descriptions have been obtained from the igraph package.

Usage

```r
V(graph)
```

Arguments

- **graph**: The graph (an igraph graph)

Value

A vertex sequence containing all vertices, in the order of their numeric vertex ids.
See Also

\texttt{V} for a complete description on this function

Examples

\begin{verbatim}
MyData <- coexpression.data
My_graph <- graph_from_data_frame(MyData)
My_graph_vertices <- V(My_graph)
\end{verbatim}
Index

*Topic IHS  
ihs, 12
*Topic association_assessment  
cond.prob.analysis, 5  
double.cent.assess, 7  
double.cent.assess.noRegression, 8
*Topic betweenness_centrality  
betweenness, 2
*Topic conditional_probability  
cond.prob.analysis, 5
*Topic datasets  
centrality.measures, 3  
coexpression.adjacency, 4  
coexpression.data, 4
*Topic degree_centrality  
degree, 6
*Topic dependence_assessment  
double.cent.assess, 7  
double.cent.assess.noRegression, 8
*Topic graph_from_adjacencymatrices  
graph_from_adjacency_matrix, 9
*Topic graph_from_dataframe  
graph_from_data_frame, 11
*Topic graph_vertices  
V, 13
*Topic integrated_hubness_score  
ihs, 12
*Topic neighborhood_connectivity  
neighborhood.connectivity, 12
ad.test, 8, 9  
adjmatrix2graph  
(graph_from_adjacency_matrix), 9
BC (betweenness), 2  
betweenness, 2, 3, 6, 13  
centrality.measures, 3  
coexpression.adjacency, 4  
coexpression.data, 4  
cond.prob.analysis, 5, 8, 9  
CPA (cond.prob.analysis), 5  
dataframe2graph  
(graph_from_data_frame), 11  
DC (degree), 6  
DCA (double.cent.assess), 7  
DCANR  
(double.cent.assess.noRegression), 8  
degree, 3, 6, 6, 13  
double.cent.assess, 5, 7, 9  
double.cent.assess.noRegression, 5, 8, 8  
gam, 8  
graph_from_adjacency_matrix, 9, 10, 11  
graph_from_data_frame, 10, 11  
hoeffd, 8, 9  
IHS (ihs), 12  
ihs, 12  
lm, 8  
NC (neighborhood.connectivity), 12  
neighborhood.connectivity, 3, 6, 12  
NNS.dep, 8, 9  
V, 13, 14  
vertices (V), 13