Package 'DNT'

February 26, 2021

Type Package

Title Differential network tests

Version 0.1.0

Author Roman Schefzik and Leonie Boland

Maintainer Roman Schefzik <roman.schefzik@medma.uni-heidelberg.de>

Description The DNT package provides a frame for testing for differences between two statistical networks based on permutation tests. Various options for network estimation (i.e. specifying the adjacency matrix) and network comparison (using several overall, node-specifc or edge-specific network difference chacteristics) are implemented. Moreover, tools for graphical illustrations and comparisons are offered.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports shiny,
igraph,
Hmisc,
ppcor,
energy,
qgraph,
stats,
dplyr,
readxl,
stringr,
tools,
utils,
shinybusy

R topics documented:

`betweenness.inv` *Calculates differences in betweenness for each network node*

Description

This function calculates the differences in betweenness for each node between two networks (adjacency matrices of the same dimension).

Usage

```
betweenness.inv(X, Y)
```

Arguments

`X, Y`	adjacency matrices of the same dimension

Details

This function calculates the differences in betweenness for each node between two networks (adjacency matrices of the same dimension). Differences in betweenness are one of the node-specific network difference characteristics to compare two networks.

Value

a vector of length N (number of nodes), containing the differences in betweenness between the two networks for each node

Examples

```
X<-create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
Y<-create.adjacency.matrix(ExDataB,methodlist=list("Spearman"))
betweenness.inv(X,Y)
```

`closeness.inv` *Calculates differences in closeness for each network node*

Description

This function calculates the differences in closeness for each node between two networks (adjacency matrices of the same dimension).

Usage

```
closeness.inv(X, Y)
```

Arguments

`X, Y`	adjacency matrices of the same dimension

Details

This function calculates the differences in closeness for each node between two networks (adjacency matrices of the same dimension). Differences in closeness are one of the node-specific network difference characteristics to compare two networks.

Value

a vector of length N (number of nodes), containing the differences in closeness between the two networks for each node

Examples

```
X<-create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
Y<-create.adjacency.matrix(ExDataB,methodlist=list("Spearman"))
closeness.inv(X,Y)
```

`comp.plot` *Provides plots allowing for visual comparison of two networks*

Description

This function provides plots allowing for visual comparison of two networks, which are given by adjacency matrices that are produced out of input data tables, using an estimation method specified by the user.

Usage

```
comp.plot(
  A,
  B,
  methodlist,
  thresh = 0.05,
  networkA = TRUE,
  networkB = TRUE,
  networkAtitle = "Network A",
  networkBtitle = "Network B",
  cluster = TRUE,
  negcol = "red",
  poscol = "blue",
  multiplier = 4,
  curved = TRUE,
  layout = igraph::layout.auto,
  vSize = 16,
  tSize = 0.8
)
```

Arguments

A, B	input data tables from which the adjacency matrices will be generated, to be provided in form of matrices, arrays, data frames or tibbles
methodlist	a list specifying the method which is used to estimate and create the adjacency matrix; see details for further information
thresh	a number between 0 and 1 (default is set to 0.05) specifying the singificance level: if the p-value corresponding to an edge weight is greater than thresh, the corresponding edge weight is not considered to be significant and thus set to zero
networkA	Boolean, indicating whether the network corresponding to the first data set shall be plotted
networkB	Boolean, indicating whether the network corresponding to the second data set shall be plotted
networkAtitle	a character, indicating the title of the first network
networkBtitle	a character, indicating the title of the second network
cluster	a Boolean, indicating whether the networks shall be plotted along with the derived communities/clusters
negcol	a character, specifying the color which is used to represent edges with a negative correlation
poscol	a character, specifying the color which is used to represent edges with a positive correlation
multiplier	a number, representing the factor with which an edge weight is multiplied in order to regulate the thickness of the drawn edges
curved	a Boolean, indicating whether the edges should be drawn using a curved or solid line
layout	a function specifying the layout of the networks; see details for possible options and further information
vSize	a number, specifying the size of the nodes
tSize	a number, specifying the node label size

Details

This function provides plots allowing for visual comparison of two networks, which are given by adjacency matrices that are produced out of input data tables, using an estimation method specified by the user. The network estimation method has to be specified in form of a list in the `methodlist` argument. Currently, the following estimation methods are supported:

- `list("Spearman")`
  Edge weights are estimated using Spearman correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression `list("Spearman")` has to be provided in the `methodlist` argument.
- `list("Spearman.adj",adjustment method)`
  Edge weights are estimated using Spearman correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression `list("Spearman.adj",adjustment method)` has to be provided in the `methodlist` argument, where `adjustment method` has to be one of the options for multple testing adjustment provided by the standard `p.adjust` R function, i.e. one of `"BH"`, `"bonferroni"`, `"BY"`, `"fdr"`, `"hochberg"`, `"holm"` or `"hommel"`.
- `list("PCSpearman")`
  Edge weights are estimated using partial Spearman correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression `list("PCSpearman")` has to be provided in the `methodlist` argument.
- `list("PCSpearman.adj",adjustment method)`
  Edge weights are estimated using partial Spearman correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression `list("PCSpearman.adj",adjustment method)` has to be provided in the `methodlist` argument, where `adjustment method` has to be one of the options for multple testing adjustment provided by the standard `p.adjust` R function, i.e. one of `"BH"`, `"bonferroni"`, `"BY"`, `"fdr"`, `"hochberg"`, `"holm"` or `"hommel"`.
- `list("DistCorr")`
  Edge weights are estimated using distance correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression `list("DistCorr")` has to be provided in the `methodlist` argument. Note that the calculations may require larger computation times, as a permutation test is involved to derive the corresponding p-values for the distance correlations.
- `list("DistCorr.adj",adjustment method)`
  Edge weights are estimated using distance correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression `list("DistCorr.adj",adjustment method)` has to be provided in the `methodlist` argument, where `adjustment method` has to be one of the options for multple testing adjustment provided by the standard `p.adjust` R function, i.e. one of `"BH"`, `"bonferroni"`, `"BY"`, `"fdr"`, `"hochberg"`, `"holm"` or `"hommel"`. Note that the calculations may require larger computation times, as a permutation test is involved to derive the corresponding p-values for the distance correlations.
- `list("EBICglasso",correlation type,tuning parameter)`
  Edge weights are estimated using the EBICglasso approach. To apply this method, the expression `list("EBICglasso",correlation type,tuning parameter)` has to be provided in the `methodlist` argument. Here, `correlation type` has to be one of the association options provided by the standard `cor` R function, i.e. one of `"kendall"`, `"pearson"` or `"spearman"`. Moreover, `tuning parameter` has to be a number specifying the EBIC tuning parameter γ. Typical choices include values between 0 and 0.5, where smaller values usually lead to a higher sensitivity in that more edges are included into the network.

Note that for EBICglasso, an additional specification of the `thresh` argument is obsolete, as it is not used for the application of the method.

The following options (functions) from the `igraph` R package are provided to specify the layout of the plots in the `layout` argument:

- `igraph::layout.auto` (default)
- `igraph::layout.circle`
- `igraph::layout.davidson.harel`
- `igraph::layout.drl`
- `igraph::layout.fruchterman.reingold`
- `igraph::layout.gem`
- `igraph::layout.graphopt`
- `igraph::layout.grid`
- `igraph::layout.kamada.kawai`
- `igraph::layout.lgl`
- `igraph::layout.mds`
- `igraph::layout.reingold.tilfort`
- `igraph::layout.star`
- `igraph::layout.svd`

Value

a plot of the two specified networks, which allows for visual comparison

Examples

```
comp.plot(ExDataA,ExDataB,methodlist=list("Spearman"))
comp.plot(ExDataA,ExDataB,methodlist=list("PCSpearman.adj","bonferroni"),
layout=igraph::layout.circle,curved=FALSE)
comp.plot(ExDataA,ExDataB,methodlist=list("EBICglasso","spearman",0.1),
layout=igraph::layout.fruchterman.reingold,curved=FALSE)
comp.plot(ExDataA,ExDataB,methodlist=list("EBICglasso","pearson",0.05),
layout=igraph::layout.star,cluster=FALSE)
```

`create.adjacency.matrix`

Creates adjacency matrix

Description

This function creates an adjacency matrix out of an input data table, using an estimation method specified by the user.

Usage

```
create.adjacency.matrix(A, methodlist, thresh = 0.05)
```

Arguments

A	input data table from which the adjacency matrix will be generated, to be provided in form of a matrix, array, data frame or tibble
methodlist	a list specifying the method which is used to estimate and create the adjacency matrix; see details for further information
thresh	a number between 0 and 1 (default is set to 0.05) specifying the singificance level: if the p-value corresponding to an edge weight is greater than thresh, the corresponding edge weight is not considered to be significant and thus set to zero

Details

This function creates an adjacency matrix out of an input data table, using an estimation method specified by the user. The network estimation method has to be specified in form of a list in the methodlist argument. Currently, the following estimation methods are supported:

- list("Spearman")
  Edge weights are estimated using Spearman correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression list("Spearman") has to be provided in the methodlist argument.
- list("Spearman.adj",adjustment method)
  Edge weights are estimated using Spearman correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression list("Spearman.adj",adjustment method) has to be provided in the methodlist argument, where adjustment method has to be one of the options for multple testing adjustment provided by the standard p.adjust R function, i.e. one of "BH", "bonferroni", "BY", "fdr", "hochberg", "holm" or "hommel".
- list("PCSpearman")
  Edge weights are estimated using partial Spearman correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression list("PCSpearman") has to be provided in the methodlist argument.
- list("PCSpearman.adj",adjustment method)
  Edge weights are estimated using partial Spearman correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression list("PCSpearman.adj",adjustment method) has to be provided in the methodlist argument, where adjustment method has to be one of the options for multple testing adjustment provided by the standard p.adjust R function, i.e. one of "BH", "bonferroni", "BY", "fdr", "hochberg", "holm" or "hommel".
- list("DistCorr")
  Edge weights are estimated using distance correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression list("DistCorr") has to be provided in the methodlist argument. Note that the calculations may require larger computation times, as a permutation test is involved to derive the corresponding p-values for the distance correlations.
- list("DistCorr.adj",adjustment method)
  Edge weights are estimated using distance correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression list("DistCorr.adj",adjustment method) has to be provided in the methodlist argument, where adjustment method has to be one of the options for multple testing adjustment provided by the standard p.adjust R function, i.e. one of "BH", "bonferroni", "BY", "fdr",

"hochberg", "holm" or "hommel". Note that the calculations may require larger computation times, as a permutation test is involved to derive the corresponding p-values for the distance correlations.

- list("EBICglasso",correlation type,tuning parameter)
  Edge weights are estimated using the EBICglasso approach. To apply this method, the expression list("EBICglasso",correlation type,tuning parameter) has to be provided in the methodlist argument. Here, correlation type has to be one of the association options provided by the standard cor R function, i.e. one of "kendall", "pearson" or "spearman". Moreover, tuning parameter has to be a number specifying the EBIC tuning parameter γ. Typical choices include values between 0 and 0.5, where smaller values usually lead to a higher sensitivity in that more edges are included into the network.
  Note that for EBICglasso, an additional specification of the thresh argument is obsolete, as it is not used for the application of the method.

Value

the adjacency matrix generated from the input data table A according to the specified estimation method

Examples

```
create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
create.adjacency.matrix(ExDataA,methodlist=list("Spearman.adj","bonferroni"))
create.adjacency.matrix(ExDataA,methodlist=list("PCSpearman"),thresh=0.1)
create.adjacency.matrix(ExDataA,methodlist=list("PCSpearman.adj","BH"),thresh=0.1)
create.adjacency.matrix(ExDataA,methodlist=list("DistCorr"))
create.adjacency.matrix(ExDataA,methodlist=list("DistCorr.adj","bonferroni"))
create.adjacency.matrix(ExDataB,methodlist=list("EBICglasso","spearman",0.1))
create.adjacency.matrix(ExDataB,methodlist=list("EBICglasso","pearson",0.05))
```

---

create.app *Creates Shiny app*

Description

Creates the Shiny app corresponding to the DNT package

Usage

```
create.app()
```

---

create.graph *Creates* igraph *graph and MST results*

Description

This function creates igraph graph and corresponding minimum spanning tree (MST) results from an adjacency matrix that is produced out of an input data table, using an estimation method specified by the user.

Usage

```
create.graph(A, methodlist, thresh = 0.05)
```

Arguments

A	input data table from which the adjacency matrix will be generated, to be provided in form of a matrix, array, data frame or tibble
methodlist	a list specifying the method which is used to estimate and create the adjacency matrix; see details for further information
thresh	a number between 0 and 1 (default is set to 0.05) specifying the singificance level: if the p-value corresponding to an edge weight is greater than thresh, the corresponding edge weight is not considered to be significant and thus set to zero

Details

This function creates igraph graph and corresponding minimum spanning tree (MST; derived using Prim's algorithm) results from an adjacency matrix that is produced out of an input data table, using an estimation method specified by the user. The function builds on respective implementations in the igraph R package.
The network estimation method has to be specified in form of a list in the methodlist argument. Currently, the following estimation methods are supported:

- list("Spearman")
  Edge weights are estimated using Spearman correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression list("Spearman") has to be provided in the methodlist argument.
- list("Spearman.adj",adjustment method)
  Edge weights are estimated using Spearman correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression list("Spearman.adj",adjustment method) has to be provided in the methodlist argument, where adjustment method has to be one of the options for multple testing adjustment provided by the standard p.adjust R function, i.e. one of "BH", "bonferroni", "BY", "fdr", "hochberg", "holm" or "hommel".
- list("PCSpearman")
  Edge weights are estimated using partial Spearman correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression list("PCSpearman") has to be provided in the methodlist argument.
- list("PCSpearman.adj",adjustment method)
  Edge weights are estimated using partial Spearman correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression list("PCSpearman.adj",adjustment method) has to be provided in the methodlist argument, where adjustment method has to be one of the options for multple testing adjustment provided by the standard p.adjust R function, i.e. one of "BH", "bonferroni", "BY", "fdr", "hochberg", "holm" or "hommel".
- list("DistCorr")
  Edge weights are estimated using distance correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression list("DistCorr") has to be provided in the methodlist argument. Note that the calculations may require larger computation times, as a permutation test is involved to derive the corresponding p-values for the distance correlations.

- `list("DistCorr.adj",adjustment method)`
  Edge weights are estimated using distance correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression `list("DistCorr.adj",adjustment method)` has to be provided in the `methodlist` argument, where `adjustment method` has to be one of the options for multple testing adjustment provided by the standard `p.adjust` R function, i.e. one of `"BH"`, `"bonferroni"`, `"BY"`, `"fdr"`, `"hochberg"`, `"holm"` or `"hommel"`. Note that the calculations may require larger computation times, as a permutation test is involved to derive the corresponding p-values for the distance correlations.
- `list("EBICglasso",correlation type,tuning parameter)`
  Edge weights are estimated using the EBICglasso approach. To apply this method, the expression `list("EBICglasso",correlation type,tuning parameter)` has to be provided in the `methodlist` argument. Here, `correlation type` has to be one of the association options provided by the standard `cor` R function, i.e. one of `"kendall"`, `"pearson"` or `"spearman"`. Moreover, `tuning parameter` has to be a number specifying the EBIC tuning parameter γ. Typical choices include values between 0 and 0.5, where smaller values usually lead to a higher sensitivity in that more edges are included into the network.
  Note that for EBICglasso, an additional specification of the `thresh` argument is obsolete, as it is not used for the application of the method.

Value

a list with 15 elements: the adjacency matrix, the `igraph` graph, the graph communities (derived by the Girvan-Newman algorithm based on edge betweenness), the graph clustering, the graph vertex degrees, the graph number of edges, the graph number of clusters, the graph number of isolated nodes, the minimum spanning tree (MST), the MST communities (derived by the Girvan-Newman algorithm based on edge betweenness), the MST clustering, the MST vertex degrees, the MST number of edges, the MST number of clusters and the MST number of isolated nodes

Examples

```
create.graph(ExDataA,methodlist=list("Spearman"))
create.graph(ExDataA,methodlist=list("Spearman.adj","bonferroni"))
create.graph(ExDataA,methodlist=list("PCSpearman"),thresh=0.1)
create.graph(ExDataA,methodlist=list("PCSpearman.adj","BH"),thresh=0.1)
create.graph(ExDataA,methodlist=list("DistCorr"))
create.graph(ExDataA,methodlist=list("DistCorr.adj","bonferroni"))
create.graph(ExDataB,methodlist=list("EBICglasso","spearman",0.1))
create.graph(ExDataB,methodlist=list("EBICglasso","pearson",0.05))
```

---

create.Igraphclustering

Creates `igraph` *graph and clustering results*

Description

This function creates `igraph` graph and clustering results from an adjacency matrix that is produced out of an input data table, using an estimation method specified by the user.

Usage

```
create.Igraphclustering(A, methodlist, thresh = 0.05)
```

Arguments

A	input data table from which the adjacency matrix will be generated, to be provided in form of a matrix, array, data frame or tibble
methodlist	a list specifying the method which is used to estimate and create the adjacency matrix; see details for further information
thresh	a number between 0 and 1 (default is set to 0.05) specifying the singificance level: if the p-value corresponding to an edge weight is greater than thresh, the corresponding edge weight is not considered to be significant and thus set to zero

Details

This function creates igraph graph and clustering results from an adjacency matrix that is produced out of an input data table, using an estimation method specified by the user. Clustering is performed using the Girvan-Newman algorithm based on edge betweenness. The function is used in the central comp.plot function of the DNT package for visual comparison of two networks and builds on respective implementations in the igraph R package.
The network estimation method has to be specified in form of a list in the methodlist argument. Currently, the following estimation methods are supported:

- list("Spearman")
  Edge weights are estimated using Spearman correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression list("Spearman") has to be provided in the methodlist argument.
- list("Spearman.adj",adjustment method)
  Edge weights are estimated using Spearman correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression list("Spearman.adj",adjustment method) has to be provided in the methodlist argument, where adjustment method has to be one of the options for multple testing adjustment provided by the standard p.adjust R function, i.e. one of "BH", "bonferroni", "BY", "fdr", "hochberg", "holm" or "hommel".
- list("PCSpearman")
  Edge weights are estimated using partial Spearman correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression list("PCSpearman") has to be provided in the methodlist argument.
- list("PCSpearman.adj",adjustment method)
  Edge weights are estimated using partial Spearman correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression list("PCSpearman.adj",adjustment method) has to be provided in the methodlist argument, where adjustment method has to be one of the options for multple testing adjustment provided by the standard p.adjust R function, i.e. one of "BH", "bonferroni", "BY", "fdr", "hochberg", "holm" or "hommel".
- list("DistCorr")
  Edge weights are estimated using distance correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression list("DistCorr") has to be provided in the methodlist argument. Note that the calculations may require larger computation times, as a permutation test is involved to derive the corresponding p-values for the distance correlations.
- list("DistCorr.adj",adjustment method)
  Edge weights are estimated using distance correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression

list("DistCorr.adj",adjustment method) has to be provided in the methodlist argument, where adjustment method has to be one of the options for multple testing adjustment provided by the standard p.adjust R function, i.e. one of "BH", "bonferroni", "BY", "fdr", "hochberg", "holm" or "hommel". Note that the calculations may require larger computation times, as a permutation test is involved to derive the corresponding p-values for the distance correlations.

- list("EBICglasso",correlation type,tuning parameter)
  Edge weights are estimated using the EBICglasso approach. To apply this method, the expression list("EBICglasso",correlation type,tuning parameter) has to be provided in the methodlist argument. Here, correlation type has to be one of the association options provided by the standard cor R function, i.e. one of "kendall", "pearson" or "spearman". Moreover, tuning parameter has to be a number specifying the EBIC tuning parameter γ. Typical choices include values between 0 and 0.5, where smaller values usually lead to a higher sensitivity in that more edges are included into the network.
  Note that for EBICglasso, an additional specification of the thresh argument is obsolete, as it is not used for the application of the method.

Value

a list with two elements: the first list element contains the results for the clustering by the Girvan-Newman algorithm based on edge betweenness; the second list element contains the results for the creation of an igraph graph

Examples

```
create.Igraphclustering(ExDataA,methodlist=list("Spearman"))
create.Igraphclustering(ExDataA,methodlist=list("Spearman.adj","bonferroni"))
create.Igraphclustering(ExDataA,methodlist=list("PCSpearman"),thresh=0.1)
create.Igraphclustering(ExDataA,methodlist=list("PCSpearman.adj","BH"),thresh=0.1)
create.Igraphclustering(ExDataA,methodlist=list("DistCorr"))
create.Igraphclustering(ExDataA,methodlist=list("DistCorr.adj","bonferroni"))
create.Igraphclustering(ExDataB,methodlist=list("EBICglasso","spearman",0.1))
create.Igraphclustering(ExDataB,methodlist=list("EBICglasso","pearson",0.05))
```

---

degree.inv *Calculates differences in degree for each network node*

Description

This function calculates the differences in degree for each node between two networks (adjacency matrices of the same dimension).

Usage

```
degree.inv(X, Y)
```

Arguments

X, Y	adjacency matrices of the same dimension

Details

This function calculates the differences in degree for each node between two networks (adjacency matrices of the same dimension). Differences in degree are one of the node-specific network difference characteristics to compare two networks.

Value

a vector of length N (number of nodes), containing the differences in degree between the two networks for each node

Examples

```
X<-create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
Y<-create.adjacency.matrix(ExDataB,methodlist=list("Spearman"))
degree.inv(X,Y)
```

---

edge.inv *Calculates the differences in absolute edge weight for each network edge*

Description

This function calculates the differences in absolute edge weight for each edge between two networks (adjacency matrices of the same dimension).

Usage

```
edge.inv(A, B)
```

Arguments

A, B	adjacency matrices of the same dimension

Details

This function calculates the differences in absolute edge weight for each edge between two networks (adjacency matrices of the same dimension). In particular, by considering absolute edge weights, it does not take account of directions (signs) of the edge weights (associations) when deriving the differences. Differences in absolute edge weight are one of the edge-specific network difference characteristics to compare two networks.

Value

a symmetric $N \times N$ matrix, with N denoting the number of nodes, containing the differences in absolute edge weight between two respective nodes

Examples

```
A<-create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
B<-create.adjacency.matrix(ExDataB,methodlist=list("Spearman"))
edge.inv(A,B)
```

edge.inv.direc *Calculates the differences in edge weight for each network edge*

Description

This function calculates the differences in edge weight for each edge between two networks (adjacency matrices of the same dimension).

Usage

```
edge.inv.direc(A, B)
```

Arguments

A, B	adjacency matrices of the same dimension

Details

This function calculates the differences in edge weight for each edge between two networks (adjacency matrices of the same dimension). In particular, it takes account of potentially different directions (signs) of the edge weights (associations) when deriving the differences. Differences in edge weight are one of the edge-specific network difference characteristics to compare two networks.

Value

a symmetric $N \times N$ matrix, with N denoting the number of nodes, containing the differences in edge weight between two respective nodes

Examples

```
A<-create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
B<-create.adjacency.matrix(ExDataB,methodlist=list("Spearman"))
edge.inv.direc(A,B)
```

eigen.inv *Calculates differences in eigenvector centrality for each network node*

Description

This function calculates the differences in eigenvector centrality for each node between two networks (adjacency matrices of the same dimension).

Usage

```
eigen.inv(X, Y)
```

Arguments

X, Y	adjacency matrices of the same dimension

Details

This function calculates the differences in eigenvector centrality for each node between two networks (adjacency matrices of the same dimension). Differences in eigenvector centrality are one of the node-specific network difference characteristics to compare two networks.

Value

a vector of length N (number of nodes), containing the differences in eigenvector centrality between the two networks for each node

Examples

```
X<-create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
Y<-create.adjacency.matrix(ExDataB,methodlist=list("Spearman"))
eigen.inv(X,Y)
```

ExDataA *Example data from which a network is constructed*

Description

example data set as a basis for constructing a network consisting of 10 nodes

Usage

```
data(ExDataA)
```

Format

a matrix consisting of 130 rows (corresponding to samples) and 10 columns (corresponding to the network nodes)

ExDataB *Example data from which a network is constructed*

Description

example data set as a basis for constructing a network consisting of 10 nodes

Usage

```
data(ExDataB)
```

Format

a matrix consisting of 130 rows (corresponding to samples) and 10 columns (corresponding to the network nodes)

frobenius.metric *Calculates the Frobenius metric*

Description

This function calculates the Frobenius metric between two adjacency matrices of the same dimension.

Usage

```
frobenius.metric(A, B)
```

Arguments

A, B	adjacency matrices of the same dimension

Details

This function calculates the Frobenius metric between two adjacency matrices of the same dimension. The Frobenius metric is an overall characteristic that can be employed to compare two networks.

Value

the Frobenius metric between the two adjacency matrices A and B

Examples

```
A<-create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
B<-create.adjacency.matrix(ExDataB,methodlist=list("Spearman"))
frobenius.metric(A,B)
```

global.str *Calculates the difference with respect to global strength*

Description

This function calculates the difference with respect to the global strength between two adjacency matrices of the same dimension.

Usage

```
global.str(A, B)
```

Arguments

A, B	adjacency matrices of the same dimension

Details

This function calculates the difference with respect to the global strength between two adjacency matrices of the same dimension. The global strength is an overall characteristic that can be employed to compare two networks.

Value

the difference with respect to the global strength between the two adjacency matrices `A` and `B`.

Examples

```
A<-create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
B<-create.adjacency.matrix(ExDataB,methodlist=list("Spearman"))
global.str(A,B)
```

`import.table` *Imports an external table into R*

Description

This function imports an external table into R.

Usage

```
import.table(directory)
```

Arguments

`directory`	the directory to the external table that should be imported into R

Details

This function imports an external table into R. It causes that only columns with integers or doubles are kept and works with .RData, .txt, .xls and .csv files as input.

Value

the imported data table

layout.func *Creates layout function*

Description

This function transforms a layout string from the igraph package into a corresponding layout function.

Usage

```
layout.func(layout)
```

Arguments

layout	the layout as a string

Details

This function transforms a layout string from the igraph package into a corresponding layout function. If the string does not represent an existing layout from the igraph package, the layout function is automatically set to layout.auto.

Value

the corresponding layout function

maximum.metric *Calculates the maximum metric*

Description

This function calculates the maximum metric between two adjacency matrices of the same dimension.

Usage

```
maximum.metric(A, B)
```

Arguments

A, B	adjacency matrices of the same dimension

Details

This function calculates the maximum metric between two adjacency matrices of the same dimension. The maximum metric is an overall characteristic that can be employed to compare two networks.

Value

the maximum metric between the two adjacency matrices A and B

Examples

```
A<-create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
B<-create.adjacency.matrix(ExDataB,methodlist=list("Spearman"))
maximum.metric(A,B)
```

`number.differences` *Calculates the differences with respect to the number of edges, clusters and isolated nodes between two networks*

Description

This function calculates the differences with respect to the number of edges, clusters (obtained by the Girvan-Newman algorithm) and isolated nodes (i.e. nodes without any edge) between two networks, for both `igraph` graphs and the corresponding minimum spanning trees (MSTs).

Usage

```
number.differences(A, B)
```

Arguments

`A, B`	output results when applying the function `create.graph` to two data tables from which the two networks are constructed

Details

This function calculates the differences with respect to the number of edges, clusters (obtained by the Girvan-Newman algorithm) and isolated nodes (i.e. nodes without any edge) between two networks, for both `igraph` graphs and the corresponding minimum spanning trees (MSTs). It builds on the `create.graph` function, which creates `igraph` graph and corresponding MST results from adjacency matrices that are produced out of input tables, using an estimation method specified by the user; see the documentation of `create.graph` for further information. Differences in numbers of edges, clusters and isolated nodes are overall characteristics that can be employed to compare two networks.

Value

a vector of length 6 containing the graph difference in the number of edges, the graph difference in the number of clusters, the graph difference in the number of isolated nodes, the MST difference in the number of edges, the MST difference in the number of clusters and the MST difference in the number of isolated nodes

Examples

```
A<-create.graph(ExDataA,methodlist=list("Spearman"))
B<-create.graph(ExDataB,methodlist=list("Spearman"))
number.differences(A,B)
```

perm.test.nw *Permutation-based test for differences between two networks*

Description

This function provides a permutation-based frame for testing for differences between two networks. In particular, various (i) network estimation methods and (ii) network difference characteristics can be specified.

Usage

```
perm.test.nw(
  A,
  B,
  permnum,
  methodlist,
  thresh = 0.05,
  score.funct,
  paired = FALSE
)
```

Arguments

A, B	input data tables from which the adjacency matrices will be generated, to be provided in form of matrices, arrays, data frames or tibbles; need to have the same number of columns (corresponding to the number of nodes)
permnum	a number, specifying the number of permutations
methodlist	a list specifying the method which is used to estimate and create the adjacency matrices; see details for possible options and further information
thresh	a number between 0 and 1 (default is set to 0.05) specifying the singificance level: if the p-value corresponding to an edge weight is greater than thresh, the corresponding edge weight is not considered to be significant and thus set to zero
score.funct	the function used to compare the adjacency matrices A and B; see details for possible options and further information
paired	Boolean, specifying whether the data underlying the two networks is paired or not

Details

This function provides a permutation-based frame for testing for differences between two networks. In particular, various (i) network estimation methods and (ii) network difference characteristics can be specified.
(i) The network estimation method has to be specified in form of a list in the methodlist argument. Currently, the following estimation methods are supported:

- list("Spearman")
  Edge weights are estimated using Spearman correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression list("Spearman") has to be provided in the methodlist argument.

- `list("Spearman.adj",adjustment method)`
  Edge weights are estimated using Spearman correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression `list("Spearman.adj",adjustment method)` has to be provided in the `methodlist` argument, where `adjustment method` has to be one of the options for multple testing adjustment provided by the standard `p.adjust` R function, i.e. one of `"BH"`, `"bonferroni"`, `"BY"`, `"fdr"`, `"hochberg"`, `"holm"` or `"hommel"`.
- `list("PCSpearman")`
  Edge weights are estimated using partial Spearman correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression `list("PCSpearman")` has to be provided in the `methodlist` argument.
- `list("PCSpearman.adj",adjustment method)`
  Edge weights are estimated using partial Spearman correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression `list("PCSpearman.adj",adjustment method)` has to be provided in the `methodlist` argument, where `adjustment method` has to be one of the options for multple testing adjustment provided by the standard `p.adjust` R function, i.e. one of `"BH"`, `"bonferroni"`, `"BY"`, `"fdr"`, `"hochberg"`, `"holm"` or `"hommel"`.
- `list("DistCorr")`
  Edge weights are estimated using distance correlation, where unadjusted p-values are employed to determine significance. To apply this method, the expression `list("DistCorr")` has to be provided in the `methodlist` argument. Note that the calculations may require larger computation times, as a permutation test is involved to derive the corresponding p-values for the distance correlations.
- `list("DistCorr.adj",adjustment method)`
  Edge weights are estimated using distance correlation, where p-values adjusted for multiple testing are employed to determine significance. To apply this method, the expression `list("DistCorr.adj",adjustment method)` has to be provided in the `methodlist` argument, where `adjustment method` has to be one of the options for multple testing adjustment provided by the standard `p.adjust` R function, i.e. one of `"BH"`, `"bonferroni"`, `"BY"`, `"fdr"`, `"hochberg"`, `"holm"` or `"hommel"`. Note that the calculations may require larger computation times, as a permutation test is involved to derive the corresponding p-values for the distance correlations.
- `list("EBICglasso",correlation type,tuning parameter)`
  Edge weights are estimated using the EBICglasso approach. To apply this method, the expression `list("EBICglasso",correlation type,tuning parameter)` has to be provided in the `methodlist` argument. Here, `correlation type` has to be one of the association options provided by the standard `cor` R function, i.e. one of `"kendall"`, `"pearson"` or `"spearman"`. Moreover, `tuning parameter` has to be a number specifying the EBIC tuning parameter γ. Typical choices include values between 0 and 0.5, where smaller values usually lead to a higher sensitivity in that more edges are included into the network.
  Note that for EBICglasso, an additional specification of the `thresh` argument is obsolete, as it is not used for the application of the method.

(ii) To quantify differences between two networks, the following (a) overall, (b) edge-specific and (c) node-specific network difference characteristics, which have to be supplied in the `score.funct` argument, are currently supported:

- `frobenius.metric` (overall)
  Calculates the Frobenius metric between two networks
- `global.str` (overall)
  Calculates the difference in global strength between two networks

- `maximum.metric` (overall)
  Calculates the maximum metric between two networks
- `number.differences` (overall)
  Calculates the differences in numbers of edges, clusters and isolated nodes between two networks
- `spec.dist` (overall)
  Calculates the spectral distance between two networks
- `betweenness.inv` (node-specific)
  Calclulates the differences in betweenness between two networks for each node
- `closeness.inv` (node-specific)
  Calculates the differences in closeness between two networks for each node
- `degree.inv` (node-specific)
  Calculates the differences in degree between two networks for each node
- `eigen.inv` (node-specific)
  Calculates the differences in eigenvector centrality between two networks for each node
- `edge.inv` (edge-specific)
  Calculates the differences in absolute edge weights between two networks for each edge
- `edge.inv.direc` (edge-specific)
  Calculates the differences in edge weights between two networks for each edge

Typically, a large number of permutations (e.g. 1000 or 10000) should be chosen in order to obtain reliable results. Note that a large number of permutations may lead to increasing computation times.

Note that when underlying data is paired (`paired=TRUE`), the input data tables need to have exactly the same dimension, i.e. the same number of columns (nodes) AND rows (samples).

Value

a list, whose specific structure depends on the specified network difference characteristic in the argument `score.funct` (with N denoting the number of nodes):

- if `score.funct` is one of `frobenius.metric`, `global.str`, `maximum.metric` or `spec.dist`:
  a list with 6 elements: the adjacency matrix for input dat set `A` ($N \times N$ matrix), the adjacency matrix for input dat set `B` ($N \times N$ matrix), the value of the test statistic (vector of length 1), the values of the test statistics when applying the permutations (vector of length `permnum`), the p-value (vector of length 1), the p-value when inserting a pseudocount in the p-value calculation to avoid p-values that are exactly zero in permutation-based settings (vector of length 1)
- if `score.funct` is `number.differences`:
  a list with 6 elements: output when applying `create.graph` to input data table `A` (list with 15 elements; see documentation of `create.graph` function for details), output when applying `create.graph` to input data table `B` (list with 15 elements; see documentation of `create.graph` function for details), the value of the test statistics (vector of length 6), the values of the test statistics when applying the permutations (`permnum`$\times 6$ matrix), the p-values (vector of length 6), the p-values when inserting a pseudocount in the p-value calculation to avoid p-values that are exactly zero in permutation-based settings (vector of length 6)
- if `score.funct` is one of `betweenness.inv`, `closeness.inv`, `degree.inv` or `eigen.inv`:
  a list with 6 elements: the adjacency matrix for input dat set `A` ($N \times N$ matrix), the adjacency matrix for input dat set `B` ($N \times N$ matrix), the value of the test statistic for each node (vector of length N), the values of the test statistics for each node when applying the permutations (`permnum`$\times N$ matrix), the p-value for each node (vector of length N), the p-value for each node when inserting a pseudocount in the p-value calculation to avoid p-values that are exactly zero in permutation-based settings (vector of length N)

- if score.funct is one of edge.inv or edge.inv.direc:
  a list with 8 elements: the adjacency matrix for input dat set A ($N \times N$ matrix), the adjacency matrix for input dat set B ($N \times N$ matrix), the value of the test statistic for each node-node pair ($N \times N$ matrix), the values of the test statistics for each node-node pair when applying the permutations (permnum$\times N \times N$ array), the p-value for each node-node pair ($N \times N$ matrix), the p-value for each node-node pair when inserting a pseudocount in the p-value calculation to avoid p-values that are exactly zero in permutation-based settings ($N \times N$ matrix), a simplified overview of the p-values for the node-node pairs ($\frac{N(N-1)}{2} \times 3$ matrix; node-node pair in columns 1 and 2, corresponding p-value in column 3), a simplified overview of the p-values for the node-node pairs when inserting a pseudocount in the p-value calculation ($\frac{N(N-1)}{2} \times 3$ matrix; node-node pair in columns 1 and 2, corresponding p-value in column 3)

Examples

```
##examples using (a) overall network difference characteristics
res1<-perm.test.nw(A=ExDataA,B=ExDataB,permnum=10000,methodlist=list("PCSpearman"),
score.funct=frobenius.metric)
res2<-perm.test.nw(A=ExDataA,B=ExDataB,permnum=10000,methodlist=list("Spearman"),
score.funct=global.str,paired=TRUE)

##examples using (b) node-specific network difference characteristics
res3<-perm.test.nw(A=ExDataA,B=ExDataB,permnum=10000,methodlist=list("Spearman"),
thresh=0.1,score.funct=betweenness.inv,paired=TRUE)
res4<-perm.test.nw(A=ExDataA,B=ExDataB,permnum=10000,methodlist=list("EBICglasso",
"spearman",0.1), score.funct=degree.inv)

##examples using (c) edge-specific network difference characteristics
res5<-perm.test.nw(A=ExDataA,B=ExDataB,permnum=10000,methodlist=list("Spearman.adj",
"bonferroni"),score.funct=edge.inv)
res6<-perm.test.nw(A=ExDataA,B=ExDataB,permnum=10000,methodlist=list("EBICglasso",
"spearman",0.01),score.funct=edge.inv.direc,paired=TRUE)
```

---

server *Server function for Shiny app*

Description

Server function for Shiny app

Usage

```
server(input, output, session)
```

Arguments

input	list-like object containing all the input data sent from the browser, named according to the input ID
output	list-like object, named according to the output ID
session	environment that can be employed to access information and functionality with respect to the session

Details

As one never calls the server function by oneself, one will never create the three argument objects by oneself. Instead, they are created by Shiny when the session starts, connecting back to a specific session.

Value

Server function for Shiny app

`spec.dist` *Calculates the spectral distance*

Description

This function calculates the spectral distance between two adjacency matrices of the same dimension.

Usage

```
spec.dist(A, B)
```

Arguments

`A, B`	adjacency matrices of the same dimension

Details

This function calculates the spectral distance between two adjacency matrices of the same dimension. The spectral distance is an overall characteristic that can be employed to compare two networks.

Value

the spectral distance between the two adjacency matrices `A` and `B`

Examples

```
A<-create.adjacency.matrix(ExDataA,methodlist=list("Spearman"))
B<-create.adjacency.matrix(ExDataB,methodlist=list("Spearman"))
spec.dist(A,B)
```

Index