Package 'BRIDES'

September 6, 2020

Type Package

Depends R (>= 3.0), igraph (>= 1.0.0), doParallel, parallel

Title BRIDES: Characterizing Path Types in Evolving Networks

Version 1.2.0

Date 2020-08-20

Author
Etienne Lord <m.etienne.lord@gmail.com>, Cindy Bouchard (UdeM), Vladimir Makarenkov (UQAM) and Francois-Joseph Lapointe (UdeM)

Maintainer Etienne Lord <m.etienne.lord@gmail.com>

Description Characterize the evolution of an original network X
into an augmented network Y by counting the number of
Breakthroughs, Roadblocks, Impasses, Detours, Equal and Shortcuts paths.

URL https://github.com/etiennelord/BRIDES

License GPL-3

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-08-26 23:23:36

R topics documented:

BRIDES-package *BRIDES: Characterize path types in evolving networks*

Description

This package allows the user to follow the evolution of an original network X into an augmented network Y by counting the number of Breakthroughs, Roadblocks, Impasses, Detours, Equal paths and Shortcuts (BRIDES) in the network Y.

Six different types of paths in Y that *pass by at least one augmented node* (*i.e.* a node that exists in Y but not in X) can be defined as follows:

1) *Breakthrough* is a path that is impossible in network X, but is possible in network Y.

2) *Roadblock* is a path that is possible in network X, but is impossible in network Y.

3) *Impasse* is a path that is impossible in both networks, X and Y.

4) *Detour* is a path that is shorter in network X than in network Y.

5) *Equal* is a path that has the same length in networks X and Y.

6) *Shortcut* is a path that is longer in network X than in network Y.

Details

Package:	BRIDES
Type:	Package
Version:	1.2.-
Date:	2020-08-15
License:	GPL-3
URL:	https://github.com/etiennelord/BRIDES
Maintainer:	Etienne Lord <m.etienne.lord@gmail.com>, François-Joseph Lapointe <francois-joseph.lapointe@umontreal.ca>

Function `BRIDES` compute the statistics between the two networks X and Y.

Function `export_network` save to a file the selected network.

Function `load_network` loads a network from a list of edges (tab-separated).

Function `save_network` and `plot_network` creates rendering of network X and network Y in various file formats (svg, png and eps).

Author(s)

Cindy Bouchard, Etienne Lord, Vladimir Makarekov and François-Joseph Lapointe

References

Lord, E., Le Cam, M., Bapteste, É., Méheust, R., Makarenkov, V., & Lapointe, F. J. (2016). BRIDES: a new fast algorithm and software for characterizing evolving similarity networks using breakthroughs, roadblocks, impasses, detours, equals and shortcuts. PloS one, 11(8), e0161474.

See Also

BRIDES, load_network, save_network export_network plot_network

---

BRIDES *Main function to calculate the evolving networks*

Description

This function calculates the different path types between two networks

Usage

```
BRIDES(X, Y = NULL, src = "default", dest = "default",
                  attributes = "default", random = 0, maxdistance = 100,
                  maxtime = 100, maxnode = 100, maxcores = 1, outfile =
                  "", size = 1000, first = 0, last = 0, weighted = TRUE,
                  runmode = "default", path_information = TRUE, verbose
                  = TRUE, min_additional = 1, max_additional = 0,
                  max_iters = 100, mutationChance = 0.1, wt = c(3, -1,
                  -2, 1, 1, 2))
```

Arguments

X	the original network X (an igraph network object)
Y	the augmented network Y with additional nodes (all the original nodes from X must be present in the augmented network Y)
src	Specify a source node (must be used with the dest argument)
dest	Specify a destination node (must be used with the src argument.)
attributes	If only one network is given, attribute which can be specified for an added nodes. Must be in the "tax" attribute of the igraph network: e.g. V(Y)$tax
first	Specify the starting path number for the BRIDES computation if distributed
last	Specify the ending path number for the BRIDES computation if distributed
min_additional	Specify for the scenarios the minimum number of additional nodes to consider
max_additional	Specify for the scenarios the maximum number of additional nodes to consider
maxdistance	Maximum distance to an added node, default: 100
max_iters	Specify the number of iterations to search for the 'genetics' runmode
maxnode	Maximum number of random additional nodes in network Y to consider
maxcores	Specify the number of compute cores (threads) to use for the calculation
maxtime	Specify the maximum time (in seconds) to individual path search, default: 100 seconds
mutationChance	Specify mutation chance for 'genetics' runmode, default=0.1
outfile	Specify an output file for the each computed paths between each pairs of nodes
path_information	Record path information (disabling this lead to faster execution), default: TRUE
random	If specified, will sample a number of random pathways

runmode	Runmode: 'default' (BRIDES statistics only) 'exhaustive' scenario-based search the maximum BRIDES score using the wt scoring scheme 'genetics' scenario-based genetics algorithm search for the best BRIDES score 'stepwise' scenario-based incremental search adding the best one node, than the second best to maximize the BRIDES score using the wt scoring scheme Note : for those scenarios, a min_additional and max_additional node must be provided Note^2: for the 'genetics' runmode, max_iters should be large enough.
size	Specify the number of paths to consider if computation is distributed
verbose	Verbose to the screen some information, default: TRUE
weighted	Specify if we consider the edge's lengths, default: TRUE
wt	Scoring sheme [B R I D E S] for the runmodes 'exhaustive','genetics','incremental'

Examples

```
## Not run:
data(networkX)
data(networkY)
set.seed(1)
###############################################################################
#           Basic usage of a BRIDES network analysis in R                     #
###############################################################################
results<-BRIDES(networkX,networkY, outfile="outfile.txt")
print(results)
# [...Results...]
#    B      R      I      D      E      S (utime stime)
    #    9      2      20     26     5      4   0.07   1.42
# Note: pathways and BRIDES statistics will be saved to outfile.txt
###############################################################################
#              Saving path information to file outfile.txt                    #
###############################################################################
BRIDES(networkX,networkY, outfile="outfile.txt")
###############################################################################
# Searching for a specific path between the source and destination nodes #
###############################################################################
BRIDES(networkX,networkY,src="x1",dest="x7")
#Prunning unconnected nodes...
#x1 -> x7
#original: 1
#augmented: 4
#[1] "x1"  "x14" "x6"  "x5"  "x7"
#[1] 0 1 0 0 0
#$`from`
#[1] "x1"
#$to
#[1] "x7"
#$path_type
#[1] "Detour"
#$path_type0
#[1] 4
#$original_path_length
#[1] 1
#$augmented_path_length
```

```
#[1] 4
#$path
#[1] "x1"  "x14" "x6"  "x5"  "x7"
#$path_visited_taxa
#[1] 0 1 0 0 0
##############################################################################
#      Sampling 100 random paths in a Erdős–Rényi random network         #
##############################################################################
set.seed(25)
#Create an original network X of 20 nodes with 5 additional in network Y
g<-random_network(20,5,type="erdos")
BRIDES(g$g1,g$g2,random=100)
##############################################################################
#    Limit the distance in network Y using the maxdistance parameter     #
##############################################################################
BRIDES(networkX,networkY, maxdistance=1)
# [...Results...]
#    B      R      I      D      E      S (utime stime)
    #    9      25     20     3      5      4   0.04   0.93  (with maxdistance=1)
    #    9      2      20     26     5      4   0.07   1.42 (without)
#           *             *
##############################################################################
#      Default analysis with an attribute file and directed network      #
##############################################################################
# Note: see https://github.com/etiennelord/BRIDES for the sample files
U0<-load_network("sample/U0.txt","sample/U0.attr.txt",directed=T)
plot_network(U0, attributes="2")
# look in vertex tax attributes for the additional nodes
results<- BRIDES(U0, attributes="2", outfile="directed.txt")
U0<-load_network("U0.txt","U0.attr.txt",directed=T)
##############################################################################
#                    Weigthed networks analysis                          #
##############################################################################
# Note: see https://github.com/etiennelord/BRIDES for the sample files
U0<-load_network("sample/U0.txt","sample/U0.attr.txt",directed=T,
                 edge_weight="proportional")
plot_network(U0, attributes="2")
results<- BRIDES(U0, attributes="2", outfile="directed.txt")
##############################################################################
#                    Weigthed networks analysis                          #
##############################################################################
# Note: if your networks already have weighted edges,
#       they will be taken into account.
data(dX)
data(dY)
plot_network(dX,dY,legend_position="topleft")
BRIDES(dX,dY, outfile="directed2.txt")
# [ BRIDES Results ]
#     B      R      I      D      E      S (utime stime)
#    5.00   3.00   4.00   0.00   0.00   0.00   0.03   0.30
# [ directed2.txt ]
# src dest dist_x dist_y BRIDES path path_attributes
# x2 x1 Inf 3 Breakthrough x2,x4,y1,x1 0,0,1,0
# x5 x1 Inf 3 Breakthrough x5,x4,y1,x1 0,0,1,0
# x4 x1 Inf 2 Breakthrough x4,y1,x1 0,1,0
# x1 x2 1 Inf Roadblock
# x5 x2 Inf 3 Breakthrough x5,x4,y1,x2 0,0,1,0
```

```
# x4 x2 Inf 2 Breakthrough x4,y1,x2 0,1,0
# x1 x5 Inf Inf Impasse
# x2 x5 Inf Inf Impasse
# x4 x5 Inf Inf Impasse
# x1 x4 2 Inf Roadblock
# x2 x4 1 Inf Roadblock
# x5 x4 Inf 1 Impasse
##############################################################################
# Scenarios: Exhaustive search                                               #
##############################################################################
# Searching for the scenario favoring:Breakthough [B=1 R=0 I=0 D=0 E=0 S=0]
# The scenario is to add 1 to 2 new nodes to network X
# The search will be un undirected and unweighted networks
# This search mode will evaluate ALL possible scenarios
# Note: this search mode could take some time on bigger networks
data(networkX)
data(networkY)
results<-BRIDES(networkX, networkY,runmode='exhaustive',max_additional=2, wt=c(1,0,0,0,0,0))
# [...Results...]
# Using scoring scheme [ B R I D E S ] : 1 0 0 0 0 0
# Exhaustive search for 6 iterations with min. : 1 and max.:2 additional nodes.
# [Iteration 1/6] (score:0)
# [Iteration 2/6] (score:9)
# [Iteration 3/6] (score:0)
# [Iteration 4/6] (score:9)
# [Iteration 5/6] (score:0)
# [Iteration 6/6] (score:9)
# Exhaustive search done.
#         B R I  D  E S X.utime stime. score best
# x14     0 4 29 25 4 4    0.05   1.19     0
# x15     9 3 20 29 3 2    0.05   1.34     9    *
# x13     0 5 29 32 0 0    0.04   1.30     0
# x14,x15 9 3 20 25 5 4    0.10   1.19     9    *
# x14,x13 0 2 29 27 4 4    0.04   1.16     0
# x15,x13 9 2 20 30 3 2    0.05   1.28     9    *
# * Best solutions found using criteria.
##############################################################################
# Scenarios: Genetics search                                                 #
##############################################################################
# Searching for the scenario favoring:Breakthough [B=1 R=0 I=0 D=0 E=0 S=0]
# The scenario is to add 1 to 2 new nodes to network X
# The search will be un undirected and unweighted networks
# This search mode will use a genetic algorithm to converge to a solution
# using artificial crossing over between the best local solutions.
# Note: It is important to set a max_iters (iteration) in this mode
# Note2: There is no garanty that a the maximum score will be evaluated
data(networkX)
data(networkY)
results<-BRIDES(networkX, networkY,runmode='genetics',max_additional=2, wt=c(1,0,0,0,0,0),
                max_iters=20)
# [...Results...]
# Using scoring scheme [ B R I D E S ] : 1 0 0 0 0 0
# Pre-training. Evaluating 3 solutions.
# Optimization for 20 iterations with min.:1 max.:2 additional nodes.
# [Iteration 1/20] (score:0)
# [Iteration 2/20] (score:9)
# [Iteration 3/20] (score:9)
```

```
# No more suitable solutions found.
# Genetics search done.
#         B R  I  D E S X.utime stime. score best
# x14     0 4 29 25 4 4    0.06   1.19     0
# x15     9 3 20 29 3 2    0.04   1.33     9    *
# x13     0 5 29 32 0 0    0.03   1.31     0
# x14,x13 0 2 29 27 4 4    0.06   1.14     0
# x15,x13 9 2 20 30 3 2    0.03   1.27     9    *
# x14,x15 9 3 20 25 5 4    0.04   1.20     9    *
# * Best solutions found using criteria.
##############################################################################
# Scenarios: Stepwise search                                                 #
##############################################################################
# Searching for the scenario favoring:Breakthough [B=1 R=0 I=0 D=0 E=0 S=0]
# The scenario is to add 1 to 2 new nodes to network X
# The search will be un undirected and unweighted networks
# This search mode add iteratively the first best node, then try to add
# the second best node, etc.
data(networkX)
data(networkY)
results<-BRIDES(networkX, networkY,runmode='stepwise',max_additional=2, wt=c(1,0,0,0,0,0))
# [...Results...]
# Using scoring scheme [ B R I D E S ] : 1 0 0 0 0 0
# Stepwise search for maximum :2 additional nodes.
# Starting adding new nodes:  1  of  2
# [Iteration 1/3] (in progress)
# [Iteration 2/3] (in progress)
# [Iteration 3/3] (in progress)
# Starting adding new nodes:  2  of  2
# [Iteration 1/2] (in progress)
# [Iteration 2/2] (in progress)
# Stepwise search done.
#         B R  I  D E S X.utime stime. score best
# x14     0 4 29 25 4 4    0.03   1.16     0
# x15     9 3 20 29 3 2    0.07   1.34     9    *
# x13     0 5 29 32 0 0    0.04   1.30     0
# x14,x15 9 3 20 25 5 4    0.04   1.19     9    *
# x15,x13 9 2 20 30 3 2    0.05   1.28     9    *
# * Best solutions found using criteria.
#
# Note that x14,x13 addition was not evaluated
##############################################################################
# Scenarios: Stepwise search (with 2,3) additional nodes                     #
##############################################################################
# The scenario is to add 1 or 2 new nodes to network X
# Searching for the scenario using the default : [B=3 R=-1 I=-2 D=1 E=1 S=2]
#
set.seed(42)
g<-random_network(20,5,type="watts") # Create a random network
plot_network(g$g1, g$g2)
BRIDES(g$g1,g$g2,runmode='stepwise',max_additional=2)
# [ Complete output ]
# Using scoring scheme [ B R I D E S ] : 3 -1 -2 1 1 2
# Stepwise search for maximum :2 additional nodes.
# Starting adding new nodes:  1  of  2
# [Iteration 1/5] (in progress)
# [Iteration 2/5] (in progress)
```

```
# [Iteration 3/5] (in progress)
# [Iteration 4/5] (in progress)
# [Iteration 5/5] (in progress)
# Starting adding new nodes:  2  of  2
# [Iteration 1/4] (in progress)
# [Iteration 2/4] (in progress)
# [Iteration 3/4] (in progress)
# [Iteration 4/4] (in progress)
# Stepwise search done.
#    B  R   I  D E S X.utime stime. score best
# x13     53 48  89  0 0 0    0.09   2.13   -67
# x17      0 48 142  0 0 0    0.13   1.67  -332
# x18      0 26 142 15 5 2    0.07   2.02  -286
# x23      0 48 142  0 0 0    0.11   1.66  -332
# x24      5 39 137  8 0 1    0.09   1.80  -288
# x13,x17 53 48  89  0 0 0    0.09   2.04   -67
# x13,x18 53 26  89 15 5 2    0.10   2.27   -21
# x13,x23 53 48  89  0 0 0    0.09   2.07   -67
# x13,x24 67 39  75  8 0 1    0.06   2.30    22    *
#* Best solutions found using criteria.
#
# [ Results list object ]
#$`population` (The evaluated scenarios)
#  [,1] [,2] [,3] [,4] [,5]
#1    1    0    0    0    0
#2    0    1    0    0    0
#3    0    0    1    0    0
#4    0    0    0    1    0
#5    0    0    0    0    1
#6    1    1    0    0    0
#7    1    0    1    0    0
#8    1    0    0    1    0
#9    1    0    0    0    1
# $brides (the resulting score)
#    B  R   I  D E S (utime stime)
# [1,] 53 48  89  0 0 0   0.09   2.13
# [2,]  0 48 142  0 0 0   0.13   1.67
# [3,]  0 26 142 15 5 2   0.07   2.02
# [4,]  0 48 142  0 0 0   0.11   1.66
# [5,]  5 39 137  8 0 1   0.09   1.80
# [6,] 53 48  89  0 0 0   0.09   2.04
# [7,] 53 26  89 15 5 2   0.10   2.27
# [8,] 53 48  89  0 0 0   0.09   2.07
# [9,] 67 39  75  8 0 1   0.06   2.30
# $size (the maximum number of additional node)
# [1] 5
# $evaluations (the score according the the wt)
# [1]  -67 -332 -286 -332 -288  -67  -21  -67   22
# $additional_nodes (the combination of additionnal nodes)
# [1] "x13" "x17" "x18" "x23" "x24" "x13,x17" "x13,x18" "x13,x23" "x13,x24"
#
# $final (the *BEST* scenario, there might be more than one)
#[1] 1 0 0 0 1
# $final_score (final scenario score)
#[1] 22
# $final_chromosome (the scenario position in the population)
#[1] 9
```

```
# $nodes (the available additional nodes)
# [1] "x13" "x17" "x18" "x23" "x24"
# $final_table (displayed table)
#   B  R   I  D E S X.utime stime. score best
# x13     53 48  89  0 0 0    0.09   2.13   -67
# x17      0 48 142  0 0 0    0.13   1.67  -332
# x18      0 26 142 15 5 2    0.07   2.02  -286
# x23      0 48 142  0 0 0    0.11   1.66  -332
# x24      5 39 137  8 0 1    0.09   1.80  -288
# x13,x17 53 48  89  0 0 0    0.09   2.04   -67
# x13,x18 53 26  89 15 5 2    0.10   2.27   -21
# x13,x23 53 48  89  0 0 0    0.09   2.07   -67
# x13,x24 67 39  75  8 0 1    0.06   2.30    22    *
#
# $networks (a list of the *BEST* resulting networks)
# $networks[[1]]
# IGRAPH c9b4e9f UNW- 22 21 -- Watts-Strogatz random graph
# + attr: name (g/c), dim (g/n), size (g/n), nei (g/n), p (g/n),
#   loops (g/l), multiple (g/l), name (v/c), tax (v/c),
# | weight (e/n)
# + edges from c9b4e9f (vertex names):
    # [...]


## End(Not run)
```

---

dX *An original directed network X with 4 nodes and 2 edges*

Description

This dataset is composed of a directed augmented sample network with one disconnected node

Usage

```
data(networkX)
```

Format

An igraph object containing network X

dX An augmented *igraph* network

Examples

```
## Not run:
data(dX)
data(dY)
plot_network(dX,dY, legend_position="topleft")
BRIDES(dX,dY)
# [Results]
#============================================================
#Total 12 pathways divided into 1 groups.
#============================================================
#Run parameters:
```

```
#Networks                   : directed
#Nodes in networkX          : 4
#Nodes in networkY          : 5
#Total added nodes (K)      : 1
#Attributes for added nodes : default
#Total paths                : 12
#Weighted edges             : FALSE
#Path information (trace)   : complete
#Group size                 : 1000
#Start group                : 1
#End group                  : 1
#Maxdistance                : 100
#Maxnode                    : 100
#Maxcores                   : 1
#=========================================================
#   B R I D E S (utime stime)
# 5 3 4 0 0 0 0.03 0.31
#====================== RESULTS ==========================
#  B      R      I      D      E      S (utime stime)
#    5.00   3.00   4.00   0.00   0.00   0.00   0.03   0.31

## End(Not run)
```

---

dY *An augmented directed network Y with 5 nodes and 7 edges*

Description

This dataset is composed of a directed augmented sample network with one additional node

Usage

```
data(networkX)
```

Format

An igraph object containing network Y

dY An augmented *igraph* network

Examples

```
## Not run:
data(dX)
data(dY)
plot_network(dX,dY, legend_position="topleft")
BRIDES(dX,dY)
# (see dX for results)

## End(Not run)
```

export_network *is an helper function to export to a text file a network*

Description

This function export to a file the given network which is compatible with Cytoscape or Gephi. One particularity is that it handle single node (disconnected) in the network.

Usage

```
export_network(g, file, attributes = FALSE)
```

Arguments

	.
	the igraph object network to export
file	the name of the file to save the network
attributes	export the igraph vertex tax attribute (i.e. V(g)$tax)), default: FALSE

Examples

```
## Not run:
data(networkX)
export_network(networkX,file="mynetworkX.txt", attributes =TRUE)

## End(Not run)
```

info_network *returns additional information regarding the networks X and Y (original and augmented).*

Description

This function returns some additional information such as number of nodes and edges in the networks X and Y.

Usage

```
info_network(g1,g2, attributes)
```

Arguments

g1	the original network X
g2	the augmented network Y with additional nodes (all the original nodes from X must be present in the augmented network Y)
attributes	the attributes

Value

Additional information such as: number of nodes in networks X and Y, clustering coefficient, average degree, number of clusters and node distributions.

Examples

```
   ## Report information about the networks of Sample_1
data(networkX)
info_network(networkX)
   ## Network characteristics:
   ## Total of new nodes in network Y: 3
   ## Number of edges in network Y: 17
   ## Number of nodes in network Y: 14
   ## Number of nodes in network X: 11
   ## Total of pathways to investigate: 55
   ## Clustering coefficient network Y: 0
   ## Clustering coefficient network X: 0
   ## Average degree + std in network Y: 2.428571 + 1.01635
   ## Average degree + std in network X: 2 + 0.8944272
   ## Average path length in network Y: 2.119403
   ## Average path length in network X: 2.162162
   ## Number of clusters in network Y: 2
   ## Number of clusters in network X: 2
   ## Average cluster size + std in network Y: 7 + 7.071068
   ## Average cluster size + std in network X: 5.5 + 4.949747
   ## Nodes distribution in network Y (first row taxa, second row count):
   ##  1  2
   ## 11  3
```

`load_network` *Helper function to load a network into igraph from a file*

Description

This function load/import a network to an*igraph* object from a file or data.frame. In particular, it can handle single node and also attributes

Usage

```
load_network(filename_or_df, filename_attributes_or_df, edge_weight,sep,header, directed)
```

Arguments

`filename_or_df`	The filename or data.frame containing the list of node
`filename_attributes_or_df`	The filename or dataframe containing annotation fo the nodes
`directed`	Do we treat the network as directed, default:FALSE
`edge_weight`	How to handle edge weights: 'equal' (unweighted, 1.0), 'proportional' (load from file), 'inverse' (1/weight)
`header`	Do the files contains header, default: FALSE
`sep`	The text separator separation columns in the file, default: 'tab'

Examples

```
## Not run:
#############################################################################
#                    Load a directed network                                #
#############################################################################
# Note: see https://github.com/etiennelord/BRIDES for the sample files
U0<-load_network("sample/U0.txt","sample/U0.attr.txt",directed=T)
plot_network(U0, attributes="2")
#############################################################################
#                    Load a weigthed network                                #
#############################################################################
U0<-load_network("sample/U0.txt","sample/U0.attr.txt",directed=T,
                  edge_weight="proportional")
plot_network(U0, attributes="2")


## End(Not run)
```

---

networkX *An original network X with 12 nodes and 11 edges*

Description

This file contains one sample network X as described in:
Lord et al. (2016). BRIDES: a new fast algorithm and software for characterizing evolving similarity networks using
breakthroughs, roadblocks, impasses, detours, equals and shortcuts. PloS one, 11(8), e0161474.

Usage

```
data(networkX)
```

Format

An igraph object containing network X

networkX An original *igraph* network

Examples

```
## Not run:
data(networkX)
data(networkY)
plot_network(networkX,networkY)
BRIDES(networkX,networkY)


## End(Not run)
#Prunning unconnected nodes...
#============================================================
#Total 66 pathways divided into 1 groups.
#============================================================
#Run parameters:
#Networks                    : undirected
```

```
#Nodes in networkX           : 12
#Nodes in networkY           : 15
#Total added nodes (K)       : 3
#Attributes for added nodes  : default
#Total paths                 : 66
#Weighted edges              : TRUE
#Path information (trace)    : complete
#Group size                  : 1000
#Start group                 : 1
#End group                   : 1
#Maxdistance                 : 100
#Maxnode                     : 100
#Maxcores                    : 1
#=========================================================
#   B R I D E S (utime stime)
# 9 2 20 26 5 4 0.07 1.27
#====================== RESULTS ============================
#     B      R      I      D      E      S (utime stime)
#  9.00   2.00  20.00  26.00   5.00   4.00   0.07   1.27
```

---

networkY *An augmented network Y with 15 nodes and 18 edges*

Description

This file contains one sample augmented network Y as described in:
Lord et al. (2016). BRIDES: a new fast algorithm and software for characterizing evolving similarity networks using
breakthroughs, roadblocks, impasses, detours, equals and shortcuts. PloS one, 11(8), e0161474.

Usage

```
data(networkX)
```

Format

An igraph object containing network y

networkY An augmented *igraph* network

Examples

```
## Not run:
data(networkX)
data(networkY)
plot_network(networkX,networkY)
BRIDES(networkX,networkY)

## End(Not run)
```

plot_network *This is an helper function to display a augmented network*

Description

This function display a representation of the given networks

Usage

```
plot_network(g1, g2 = NULL, layout = layout.kamada.kawai, taxnames
                  = "", main = "", attributes = "", legend_position =
                  "bottomright", label_filename = "")
```

Arguments

g1	the original network X
g2	the augmented network Y
main	the title of the picture
layout	*igraph* layout function (default=layout.kamada.kawai)
attributes	the attributes of the nodes added to the original graph. By default, we select all nodes that are not in g1.
taxnames	select nodes with belongs to a specfic group (in the V(g)$tax) Note that in order to display all the different groups, you must use 'allgroup' as the *taxnames* argument
legend_position	position of the legend, default: "bottomright"
label_filename	currently in development.

Examples

```
## Not run:
data(networkX)
data(networkY)
plot_network(networkX, networkY, legend_position="topleft")
#


## End(Not run)
```

random_network *creates random augmented networks X and Y*

Description

This function allows the user to create random network X and an associated augmented network Y using either the Erdos-Renyi model or the Barabasi-Albert model.

Usage

```
random_network(original_node, additional_node, ngroup, edge_ratio,total_edge, type)
```

Arguments

original_node	the number of nodes in the original network X (default=25)
additional_node	the number of additional node in network Y (default=5)
ngroup	the number of additional taxa groups in network Y (default=1)
edge_ratio	the edge to node ratio (default=between 1 and 5)
total_edge	the number of edges in the fixed model
type	Either 'erdos' for the Erdos-Renyi model, 'barabasi' for the Barabasi-Albert model, 'watts' for the Watts-Strogatz model or 'fixed' fixed model allowing a defined number of edges

Value

Return a data.frame containing *g1*, *g2*, the *total_nodes* and the *total_edges* numbers of network Y and the *total_original_nodes* number of network X.

Examples

```
## Create a small random_network
## Not run:
random_network()

## End(Not run)
## Expected result:
#
#  $g1
#   IGRAPH UNW- 25 23 -- Erdos renyi (gnm) graph
#   + attr: name (g/c), type (g/c), loops (g/l), m (g/n), name (v/c), tax
#    (v/c), weight (e/n)
#
#  $g2
#   IGRAPH UNW- 30 30 -- Erdos renyi (gnm) graph
#   + attr: name (g/c), type (g/c), loops (g/l), m (g/n), name (v/c), tax
#    (v/c), weight (e/n)
#
#   $total_nodes
# [1] 30
#
# $total_edges
#  [1] 30
#
#   $total_original_nodes
#   [1] 25
#
## Create two networks using the Erdos-Renyi model with 100 nodes in network X
## and 10 additional nodes in network Y of 3 types.
random_network(100,10,3);
## Create a random networks of 20 +10 additional node using the  Barabsi-Albert model
## and compute the corresponding SDDE path types.
## l <- random_network(20,10,ngroup=1,vertex_ratio=1, type='barabasi');
```

```
## complete_network(l$g1, l$g2);
```

---

save_network *This is an helper function to save an illustration of some network(s) to a file*

Description

This function saves to a file a representation of the given networks

Usage

```
save_network(g1,g2,filename,layout,attributes,mode,imagesize,
       main, legend_position, label_filename)
```

Arguments

g1	the original network X
g2	the augmented network Y with additional nodes (all the original nodes from X must be present in the augmented network Y)
filename	the name of the file to save the network. Note that the file extension will automatically be added
layout	*igraph* layout function (default=layout.kamada.kawai)
main	the title of the picture
attributes	the taxon name of the nodes added to the original graph. By default, we select all nodes that are not in g1. Note that in order to display all the different attributes groups, you must use 'allgroup' as the *atrtibutes* argument
mode	type of file to create either: 'png', 'svg', or 'eps' (default='png')
imagesize	size of the image in pixels (default=800 px)
legend_position	position of the legend, default: "bottomright"
label_filename	currently in development.

Examples

```
## Not run:
data(Sample_1)
## Save the networks to file using the default layout (layout.kamada.kawai)
## and size (1200px)
save_network(g1,g2,filename="sample1",imagesize=1200)
## Save the networks to a file using the layout.fruchterman.reingold layout,
## and in 'svg' format
save_network(g1,g2,filename="s", mode="svg",layout=layout.fruchterman.reingold)
## Save a network containing different groups
gsample <- random_network(ngroup=4)
save_network(gsample$g1,gsample$g2,filename="gsample", attributes="allgroup")

## End(Not run)
```

U0 *A sample directed and weighted network with node attributes*

Description

This dataset of a sample network with 8 nodes with attributes, and 9 directed and weighted edges.

Usage

```
data(U0)
```

Format

An igraph object containing network U0

U0 An augmented *igraph* network. Nodes attributes are 1 for the original network, and 2 for the augmented network.

Examples

```
## Not run:
data(U0)
plot_network(U0,attributes="2", legend_position="topleft")
BRIDES(U0, attributes="2")


## End(Not run)
```

Undocumented functions *Undocumented functions*

Description

The following functions are for internal computation only: *'all_combn' 'allsolution' 'create_sample_path' 'fixed_combn' 'good_path2' 'match_matrix' 'multicore' 'optimize_ga' 'pathBRIDES' 'save_network_big' 'split_sample' 'stepwise'* and should not be used independantly. Most of them are helper function in the calculation of the BRIDES statistics.

Index