Package 'SDDE'

August 26, 2015

Type Package

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

Title Shortcuts, Detours and Dead Ends (SDDE) Path Types in Genome Similarity Networks

Version 1.0.1

Date 2015-08-20

Author Etienne Lord, Margaux Le Cam, Eric Bapteste, Vladimir Makarenkov and Francois-Joseph Lapointe

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

Description Compares the evolution of an original network X to an augmented network Y by counting the number of Shortcuts, Detours, Dead Ends (SDDE), equal paths and disconnected nodes.

License GPL-3

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-08-26 23:23:36

R topics documented:

`SDDE-package` *Shortcuts, Detours and Dead Ends (SDDE) Path Types in Genome Similarity Networks*

Description

SDDE compares the evolution of an original network X to an augmented network Y by counting the number of shortcuts, detours, dead ends, equal paths and disconnected nodes.

Four 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) *Shortcut* is a path between two nodes, originally present in X, that is shorter in Y than in X.

2) *Detour* is a path such that it connects two nodes originally present in X and is longer in Y than in X.

3) *Dead End* is a path that exists in X, but is impossible in Y.

4) *Equal* is a path that has the same length in X an Y.

The numbers of *disconnected* nodes is also reported.

Details

Package:	SDDE
Type:	Package
Version:	1.0.1
Date:	2015-08-20
License:	GPL-2
Maintainer:	Etienne Lord <m.etienne.lord@gmail.com>, François-Joseph Lapointe <francois-joseph.lapointe@umontreal.ca>

Function complete_network computes the number of paths in each category.

Function complete_restart is similar to the complete_network function but it identifies the paths in a predefined order.

Function complete_trace is similar to the complete_network function but allow the enumeration of nodes and taxa in a path.

Function sample_network computes the number of paths in each category for a given set of nodes of the original network X.

Function random_network creates a random network X and an augmented network Y using Erdos-Renyi or Barabási–Albert models.

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

Function load_network loads a network from a list of edges (tab-separated) which is compatible with the complete_network, complete_restart and sample_network function.

Author(s)

Etienne Lord, Margaux Le Cam, Éric Bapteste, Vladimir Makarenkov and François-Joseph Lapointe

See Also

complete_network, complete_restart, complete_trace, sample_network, load_network, save_network plot_network

---

`complete_network` *compare two given networks (original and augmented, presented as undirected graphs) using a path analysis*

Description

This function computes the number of paths in each category for all pairs of nodes of the original network.

Usage

```
complete_network(g1,g2,
     taxnames,
     maxdistance=0,
     maxtime=3600,
     maxnode=0,
     verbose=FALSE,
     file="log.txt",
    maxcores=1,
    node1,
    node2
)
```

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)
taxnames	the taxon name of the nodes added to the original graph (default: we select all nodes that are not in g1)
maxnode	the maximum number of augmented nodes in network Y to take into account (default=0, no maximum number of augmented nodes to take into account. The augmented nodes are sorted by distance to the investigated node pairs by the algorithm.)
maxtime	the maximum search time per pathway (default=3600 seconds)
maxdistance	the maximum search distance to the added nodes in network Y (default=0, no maximum distance for augmented nodes to take into account)
verbose	flag to save into a file additionnal informations regarding the obtained paths (default=FALSE, the file name should be indicated)
file	filename to save the additionnal informations
node1	if node1 is set,we look only for paths starting from node1
node2	if node2 is set, we will only look at the paths starting at node1 and ending at node2
maxcores	maximum number of cores to use (default=1, use 0 to use half of the total cores available)

Value

This function returns the number of paths of each category: *Shortcuts*, *Detours*, *Dead ends*, *Equal* paths and disconnected nodes.

If some of the augmented nodes are not visited because of the limits defined by maxnode, maxtime or maxdistance, a path can also be classified as *Detour or Dead End*.

Examples

```
## Searching the sample data (containing 11 original nodes and 3 augmented nodes)
data(Sample_1)
result <- complete_network(g1, g2)
print(result)
## Results:
##
##          disconnected nodes shortcuts equals detours dead ends
##user.self                 18         4      5      26         2
##          Dead ends or detour total user time system time real time
##user.self                   0    55     0.997       0.111     2.186
##
## Searching for path x1 -> x2 in the sample data
## Not run: complete_network(g1, g2,node1="x1", node2="x2")
##
## Specifying a limit on time (60 seconds) and maximum distance (2) of
```

```
## an original node to an augmented node
## Not run: complete_network(g1, g2, maxtime=60, maxdistance=2)
```

`complete_restart` *compares two given networks (original and augmented, presented as undirected graphs) using a path analysis*

Description

This function is equivalent to the `complete_network` function, but it uses a fixed number of pathways to be investigated, which are divided in groups to allow restarting or better dispatching of tasks in a cluster environment.

Usage

```
complete_restart(g1,g2,
     taxnames,
     resultfile='result.txt',
     size=1000,
     start=1,
     end=0,
     maxdistance=0,
     maxtime=3600,
     maxnode=0,
     verbose=FALSE,
     file='log.txt',
     maxcores=1
)
```

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)
`taxnames`	the taxon name of the nodes added to the original graph (default: we select all nodes that are not in g1)
`resultfile`	filename to which results are saved after each group (default='result.txt')
`size`	the number of paths to investigate at each iteration of the algorithm (default=1000)
`start`	the starting group of path lengths to investigate (default=1)
`end`	the ending group of path lengths to investigate (default=0 *i.e.* run until the end)
`maxnode`	the maximum number of augmented nodes in network Y to take into account (default=0, no maximum number of augmented nodes to take into account)
`maxtime`	the maximum search time per pathway (default=3600 seconds)
`maxdistance`	the maximum search distance to the added nodes in network Y (default=0, no maximum distance for augmented nodes to take into account)

verbose	flag to save into a file additionnal informations regarding the obtained paths (default=FALSE, the file name should be indicated)
file	filename to save the additionnal informations
maxcores	maximum number of cores to use (default=1, use 0 to use half of the total cores available)

Value

This function returns the number of paths of each category : *Shortcuts*, *Detours*, *Dead ends*, *Equal* paths and disconnected nodes.
If some of the augmented nodes are not visited because of the limits defined by maxnode, maxtime or maxdistance, a path can also be classified as 'Detour or Dead End'.

The format of the resultfile is as follow (tab-separated-value):
Columns:
1. Group number
2. Disconnected nodes
3. Shortcuts
4. Egals
5. Detours
6. Dead ends
7. Undefined detour or dead end*
8. Total path evaluated
9. User time (seconds)
10. System time (seconds)
11. Real time (seconds)
12. Disconnected total (up to this group)
13. Shortcut total (up to this group)
14. Egal total (up to this group)
15. Detour total (up to this group)
16. Dead end total (up to this group)
17. Undefined total (up to this group)
18. Total path evaluated (up to this group)
19. Total user time (up to this group)
20. Total system time (up to this group)
21. Total real time (up to this group)

Examples

```
## Searching the sample data (containing 11 original nodes and 3 augmented nodes)
data(Sample_1)
## Run from group number 1 to 2
complete_restart(g1,g2,size=5, start=1,end=2)
```

`complete_trace` *return properties of a single path in two given networks (original and augmented, presented as undirected graphs) using a path analysis*

Description

This function is equivalent to the `complete_network` function, but it analyses a single path returning information about the: path type, the enumeration of traversed nodes, the traversed node taxa, the original path length and the augmented path length.

Usage

```
complete_trace(g1,g2,
  taxnames,
  node1,
  node2,
  maxdistance=0,
  maxtime=3600,
  maxnode=0,
  maxcores=1
)
```

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)
`taxnames`	the taxon name of the nodes added to the original graph (default: we select all nodes that are not in g1)
`node1`	the *from* node (either the node name or the node number) of the path to investigate
`node2`	the *to* node (either the node name or the node number) of the path to investigate
`maxnode`	the maximum number of augmented nodes in network Y to take into account (default=0, no maximum number of augmented nodes to take into account. The augmented nodes are sorted by distance to the investigated node pairs by the algorithm.)
`maxtime`	the maximum search time per pathway (default=3600 seconds)
`maxdistance`	the maximum search distance to the added nodes in network Y (default=0, no maximum distance for augmented nodes to take into account)
`maxcores`	maximum number of cores to use (default=1, use 0 to use half of the total cores available)

Value

This function returns a list containing the *from* and *to* node names, the *path type* (*Shortcut*, *Detour*, *Dead end* and *Equal* paths or *disconnected* node), the *original path length*, the *augmented path length*, the traversed *path* nodes and the *path_visited_taxa* (taxa of each visited node).

Examples

```
## Searching the sample data for the path type between node x1 and x5
data(Sample_1)
##
## Not run: complete_trace(g1,g2,node1="x1", node2="x5")
## Expected results:
## $from
## [1] "x1"
## $to
## [1] "x5"
## $path_type
## [1] "Detour"
## $original_path_length
## [1] 2
## $augmented_path_length
## [1] 3
## $path
## [1] "x1" "x14" "x6" "x5"
## $path_visited_taxa
## [1] 1 2 1 1
```

Eukaryote_nonphoto *An original and an augmented real genomic networks*

Description

Those networks are composed of 145 archaea, 181 eubacteria with the addition of 19 non-photosynthetic eukaryotes to illustrate the functions of this package.

Usage

```
data(Eukaryote_nonphoto)
```

Format

A data frame with the following 2 variables:

g1 An original *igraph* network

g2 An augmented *igraph* network

Examples

```
## Not run:
data(Eukaryote_nonphoto)
info_network(g1,g2)

## End(Not run)
## Network characteristics:
## Total of new nodes in network Y: 19
## Number of edges in network Y: 1845
## Number of nodes in network Y: 345
## Number of nodes in network X: 326
## Total of pathways to investigate: 52975
## Clustering coefficient network Y: 0.8655372
## Clustering coefficient network X: 0.8643119
## Average degree + std in network Y: 10.69565 + 8.366434
## Average degree + std in network X: 10.45399 + 8.502816
## Average path length in network Y: 5.287595
## Average path length in network X: 4.915441
## Number of clusters in network Y: 22
## Number of clusters in network X: 22
## Average cluster size + std in network Y: 15.68182 + 28.92583
## Average cluster size + std in network X: 14.81818 + 25.1711
## Nodes distribution in network Y (first row taxa, second row count):
##           archaea         eubacteria eukaryote_nonphoto
##               145                181                 19
```

Eukaryote_photo *An original and an augmented real genomic networks*

Description

Those networks are composed of 145 archaea, 181 eubacteria with the addition of 19 photosynthetic eukaryotes to illustrate the functions of this package.

Usage

```
data(Eukaryote_photo)
```

Format

A data frame with the following 2 variables:

g1 An original *igraph* network

g2 An augmented *igraph* network

Examples

```
## Not run:
data(Eukaryote_photo)
info_network(g1,g2)

## End(Not run)
## Network characteristics:
## Total of new nodes in network Y: 19
## Number of edges in network Y: 2014
## Number of nodes in network Y: 345
## Number of nodes in network X: 326
## Total of pathways to investigate: 52975
## Clustering coefficient network Y: 0.866758
## Clustering coefficient network X: 0.8643119
## Average degree + std in network Y: 11.67536 + 9.600091
## Average degree + std in network X: 10.45399 + 8.502816
## Average path length in network Y: 5.972801
## Average path length in network X: 4.915441
## Number of clusters in network Y: 21
## Number of clusters in network X: 22
## Average cluster size + std in network Y: 16.42857 + 32.51857
## Average cluster size + std in network X: 14.81818 + 25.1711
## Nodes distribution in network Y (first row taxa, second row count):
##        archaea      eubacteria eukaryote_photo
##            145             181              19
```

---

g1 *generic name for network X*

Description

This is the generic name for the original network X currently in use by the functions of this package.

---

g2 *generic name for network Y*

Description

This is the generic name for the augmented network Y currently in use by the functions of this package.

`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, taxnames)
```

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)
`taxnames`	the taxon name of the nodes added to the original graph (default: we select all nodes that are not in g1)

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(Sample_1)
info_network(g1,g2)
   ## 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
```

`info_node` *returns additional information regarding the nodes of networks X and Y (original and augmented)*

Description

This function returns some additional information regarding the nodes of networks X and Y such as the reachability of nodes in networks X to the augmented nodes of network Y.

Usage

```
info_node(g1,g2,taxnames, maxnode, maxdistance)
```

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)
`taxnames`	the taxon name of the nodes added to the original graph (default: we select all nodes that are not in g1)
`maxnode`	the maximum number of augmented nodes in network Y to take into account (default=0, no maximum number of augmented nodes to take into account. The augmented nodes are sorted by distance to the investigated node pairs by the algorithm.)
`maxdistance`	the maximum search distance to the added nodes in network Y (default=0, no maximum distance for augmented nodes to take into account)

Value

This function return information about the reachability of each node such as : *reach_count*(number of reachable augmented nodes), *reach_max_dist* (maximal distance to an augmented node), *reach_min_dist* (minimal distance to an augmented node), *reach_mean_dist* (mean distance to augmented nodes)

Examples

```
   ## Report reachability informations about the node in sample1
   ## Not run:
   data(Sample_1)
   info_node(g1,g2)

## End(Not run)
   ## Results:
   ## Total of pathways to investigate: 55
   ## Number of edges in network Y: 17
```

```
## Number of nodes in network Y: 14
## Number of nodes in network X: 11
## Nodes distribution in network Y (first row taxa, second row count):
## 1  2
## 11  3
## $reach_count
## [1] 3 3 3 3 3 3 0 3 3 0 3 3 3 3 3 0 3 3 0 3 3 3 3 0 3 3 0 0 3 3 0 3 3 0 3 3 0 3
## [39] 3 0 3 0 3 3 0 0 3 3 0 0 0 0 3 0 0
##  $reach_max_dist
## [1] 3 3 3 3 3 1 0 3 1 0 3 3 3 1 1 0 3 1 0 3 3 3 3 0 3 3 0 0 3 1 0 3 1 0 3 1 0 3
## [39] 1 0 1 0 3 1 0 0 1 1 0 0 0 0 1 0 0
## $reach_min_dist
## [1] 2 1 2 2 2 2 0 2 2 0 1 1 1 2 3 0 2 3 0 1 1 1 2 0 1 2 0 0 2 2 0 2 2 0 2 2 0 2
## [39] 2 0 2 0 2 2 0 0 2 2 0 0 0 0 2 0 0
## $reach_mean_dist
##  [1] 2 2 2 2 2 2 0 2 2 0 2 2 2 2 2 0 2 2 0 2 2 2 2 0 2 2 0 0 2 2 0 2 2 0 2 2 0 2
## [39] 2 0 2 0 2 2 0 0 2 2 0 0 0 0 2 0 0
```

---

load_network *is a helper function to load networks from files*

Description

This function allows the user to load the network files. Note that standard *igraph* functions can be also used.

Usage

```
load_network(filename_or_df, filename_tax_or_df, edge_weight)
```

Arguments

filename_or_df	the file containing an undirected list of nodes in tabular format (node1, node2, edge weight)
filename_tax_or_df	an additional file containing taxa information for each node
edge_weight	how to treat the edge weight: 'equal', 'proportional' or 'inverse' (default: equal)

Value

This function return generic *igraph* objects suitable for use in the SDDE package.

Examples

```
## Load a network from files.
## Not run:
network=load_network('graph.txt','graph_tax.txt','equal')
info_network(network$g1,network$g2);
```

```
## End(Not run)
# We expect the network to be a series of nodes as tab-separated values.
#
# Example graph.txt:
# node1 node2  edge weight
# x1 x2     1
# x2 x3 1
# x3 x6 1
# x1 x5 1
# x5 x6 1
#
# Example graph_tax.txt
# x1 plasmid
# x2 plasmid
# x3 bacteria
# x4 plasmid
# x5 bacteria
# x6 virus
```

`Plasmids` *An original and an augmented real genomic networks*

Description

These networks include 152 archaea, 217 eubacteria with the addition of 3 479 plasmids to illustrate the application of the SDDE package

Usage

```
data(Plasmids)
```

Format

A data frame with the following 2 variables:

g1 An original *igraph* network

g2 An augmented *igraph* network

Examples

```
## Not run:
data(Plasmids)
info_network(g1,g2)

## End(Not run)
## Network characteristics:
## Total of new nodes in network Y: 3522
## Number of edges in network Y: 187848
## Number of nodes in network Y: 3848
## Number of nodes in network X: 326
```

```
## Total of pathways to investigate: 52975
## Clustering coefficient network Y: 0.5591526
## Clustering coefficient network X: 0.8643119
## Average degree + std in network Y: 97.6341 + 119.6407
## Average degree + std in network X: 10.45399 + 8.502816
## Average path length in network Y: 2.903052
## Average path length in network X: 4.915441
## Number of clusters in network Y: 74
## Number of clusters in network X: 22
## Average cluster size + std in network Y: 52 + 417.644
## Average cluster size + std in network X: 14.81818 + 25.1711
## Nodes distribution in network Y (first row taxa, second row count):
##   archaea eubacteria    plasmid
##       152        217       3479
```

`plot_network` *Helper function to display an illustration of a network.*

Description

This function display a representation of the networks.

Usage

```
plot_network(g1,g2,layout,taxnames)
```

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)
`layout`	*igraph* layout function (default=layout.kamada.kawai)
`taxnames`	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 taxa groups, you must use 'allgroup' as the *taxnames* argument

---

`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);
```

Sample_1 *An original network X with 11 nodes and an augmented network Y of 14 nodes*

Description

This file contains two artificial networks used to illustrate the application of the SDDE package.

Usage

```
data(Sample_1)
```

Format

A data frame with 11 nodes for network g1 and 14 observations for network g2:

g1 An original *igraph* network

g2 An augmented *igraph* network

Examples

```
## Not run:
data(Sample_1)
complete_network(g1,g2);

## End(Not run)
##        disconnected nodes shortcuts equals detours dead ends
##                         18         4      5      26         2
##        Dead ends or detour total user time system time real time
##                           0    55     0.807       0.081     1.889
## (log.txt with the verbose option)
# Total new nodes in g2: 3
# Number of edges in g2: 17
# Number of nodes in g2: 14
# Number of nodes in g1: 11
# ===================================
# x3 x1 Detour
# x4 x2 Detour
# x2 x1 Detour
# x7 x1 Detour
# x7 x3 Detour
# x3 x2 Detour
# x5 x2 Detour
```

```
# x4 x1 Detour
# x4 x3 Detour
# x5 x4 Detour
# x7 x2 Detour
# x9 x1 Equal
# x7 x4 Dead
# x9 x3 Detour
# x5 x1 Detour
# x9 x7 Detour
# x10 x2 Disconnected nodes
# x10 x4 Disconnected nodes
# x10 x5 Disconnected nodes
# x5 x3 Detour
# x6 x1 Equal
# x6 x3 Detour
# x5 x7 Detour
# x9 x2 Equal
# x6 x7 Detour
# x6 x9 Detour
# x8 x2 Shortcut
# x9 x4 Detour
# x8 x4 Equal
# x9 x5 Detour
# x10 x1 Disconnected nodes
# x10 x3 Disconnected nodes
# x10 x7 Disconnected nodes
# x10 x9 Disconnected nodes
# x6 x2 Detour
# x8 x5 Detour
# x8 x10 Disconnected nodes
# x11 x1 Disconnected nodes
# x11 x3 Disconnected nodes
# x11 x7 Disconnected nodes
# x11 x9 Disconnected nodes
# x11 x6 Disconnected nodes
# x6 x4 Detour
# x6 x5 Detour
# x6 x10 Disconnected nodes
# x8 x1 Shortcut
# x8 x3 Shortcut
# x8 x7 Equal
# x8 x9 Detour
# x8 x6 Shortcut
# x11 x2 Disconnected nodes
# x11 x4 Disconnected nodes
# x11 x5 Disconnected nodes
# x11 x10 Dead end
# x11 x8 Disconnected nodes
# ====================================
```

`Sample_2` *An network X of 6 nodes and an augmented network Y with 7 nodes*

Description

This file contains two artificial networks used to illustrate the application of the SDDE package.

Usage

```
data(Sample_2)
```

Format

A data frame with 6 nodes for network g1 and 7 nodes for network g2:

g1 An original *igraph* network

g2 An augmented *igraph* network

Examples

```
## Not run:
data(Sample_2)
complete_network(g1,g2)

## End(Not run)
##          disconnected nodes shortcuts equals detours dead ends
##                           0         0      1       8         6
##          Dead ends or detour total user time system time real time
##                            0    15     0.656       0.049     1.023
# Total new nodes in g2: 1
# Number of edges in g2: 7
# Number of nodes in g2: 7
# Number of nodes in g1: 6
# ===================================
# x2 x1 Dead
# x3 x2 Dead
# x4 x2 Dead
# x5 x2 Detour
# x3 x1 Dead
# x5 x4 Detour
# x6 x2 Detour
# x4 x1 Dead
# x6 x4 Detour
# x4 x3 Dead
# x5 x1 Detour
# x5 x3 Detour
# x6 x1 Detour
# x6 x3 Detour
# x6 x5 Equal
#====================================
```

`sample_network` *compares two networks using a path analysis of total pathways*

Description

This function allows the user to compare two related networks (undirected graphs) by computing the number of paths of each category for a sample of nodes from the original graph.

Usage

```
sample_network(g1,g2,
   size,
   taxnames,
   maxdistance=0,
   maxtime=3600,
   maxnode=0,
   verbose=FALSE,
   file='log.txt',
   maxcores=1,
   node1,
   node2,
   sample_paths=c(),
   old_path=c()
)
```

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)
`taxnames`	the taxon name of the nodes added to the original graph (default: we select all nodes that are not in g1)
`size`	the number of pathways to sample or the percent of node to sample if smaller than 1.0 (default=10)
`maxnode`	the maximum number of augmented nodes in network Y to take into account (default=0, no maximum number of augmented nodes to take into account. The augmented nodes are sorted by distance to the investigated node pairs by the algorithm.)
`maxtime`	the maximum search time per pathway (default=3600 seconds)
`maxdistance`	the maximum search distance to the added nodes in network Y (default=0, no maximum distance for augmented nodes to take into account)
`verbose`	flag to save into a file additionnal informations regarding the obtained paths (default=FALSE, the file name should be indicated)
`file`	filename to save the additionnal informations

node1	if node1 is set,we look only for paths starting from node1
node2	if node2 is set, we will only look at the paths starting at node1 and ending at node2
maxcores	maximum number of cores to use (default=1, use 0 to use half of the total cores available)
sample_paths	vector containing a list of node numbers to sample. Note: this override the size argument
old_path	vector containing a list of node numbers already sampled

Value

This function returns the number of paths of each category: Shortcuts, Detours, Dead ends, Equal paths or disconnected nodes.
If not all augmented nodes are visited because of the maxnode, maxtime or maxdistance limits, a path can also be classified as 'Detour or Dead End'.

Examples

```
## Sample 10 of the 55 pathways in Sample_1
data(Sample_1)
sample_network(g1,g2, size=10)
## Repeated sampling (5) of 10 pathways
## Not run:
old_path <- c()
for (i in 1:5) {
       sample_paths <- sample_path(10, length(V(g1)), old_path=old_path);
       sample_network(g1,g2, sample_paths=sample_paths);
       old_path <- c(old_path,sample_paths);
 }

## End(Not run)
```

---

sample_path *is a helper function that creates a vector of non-repeating pathways to investigate*

Description

This internal function create a randomized vector of paired node numbers of a given length without repeat.

Usage

```
sample_path(n, max_len, old_path)
```

Arguments

n	the number of pathways
max_len	the total number of nodes
old_path	vector of already sampled pairs of node numbers

Value

This function return a randomize vector of 2 x *n* node numbers.

Examples

```
## Draw 5 samples of 10 pathways out of 100 nodes without repeated pathways
## Not run:
old_path <- c();
for (i in 1:5) {
r <- sample_path(10,100, old_path);
old_path <- c(old_path, r);
}

## End(Not run)
```

---

save_network *is an helper function to save an illustration of a network into a file*

Description

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

Usage

```
save_network(g1,g2,filename,layout,taxnames,mode,imagesize)
```

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)
taxnames	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 taxa groups, you must use 'allgroup' as the *taxnames* argument
mode	type of file to create either: 'png', 'svg', or 'eps' (default='png')
imagesize	size of the image in pixels (default=800 px)

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", taxnames="allgroup")

## End(Not run)
```

---

save_network_big *Helper function to save an illustration of a network to file with more than 50 nodes.*

Description

This function save to file a representation of the networks. Note that it will be automatically called by the save_network function.

Usage

```
save_network_big(g1,g2,filename,layout,taxnames,mode,imagesize)
```

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)
taxnames	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 taxa groups, you must use 'allgroup' as the *taxnames* argument
mode	type of file to create either: 'png', 'svg', or 'eps' (default='png')
imagesize	size of the image in pixels (default=2500 px)

SDDE.version *Version information.*

Description

This function return the current SDDE version.

Usage

```
SDDE.version()
```

Viruses *An original and an augmented real genomic networks*

Description

These networks include 145 archaea, 181 eubacteria with the addition of 1658 viruses to illustrate the application of the SDDE package.

Usage

```
data(Viruses)
```

Format

A data frame with the following 2 variables.

g1 An original *igraph* network

g2 An augmented *igraph* network

Examples

```
## Not run:
data(Viruses)
info_network(g1,g2)

## End(Not run)
## Network characteristics:
## Total of new nodes in network Y: 1658
## Number of edges in network Y: 12054
## Number of nodes in network Y: 1984
## Number of nodes in network X: 326
## Total of pathways to investigate: 52975
## Clustering coefficient network Y: 0.8010996
## Clustering coefficient network X: 0.8643119
## Average degree + std in network Y: 12.15121 + 15.62558
## Average degree + std in network X: 10.45399 + 8.502816
```

```
## Average path length in network Y: 5.233931
## Average path length in network X: 4.915441
## Number of clusters in network Y: 320
## Number of clusters in network X: 22
## Average cluster size + std in network Y: 6.2 + 20.8493
## Average cluster size + std in network X: 14.81818 + 25.1711
## Nodes distribution in network Y (first row taxa, second row count):
##   archaea eubacteria      virus
##       145        181       1658
```

Index