Package 'RandomWalkRestartMH'

October 14, 2021

Type Package

Title Random walk with restart on multiplex and heterogeneous Networks

Version 1.12.0

Date 2018-03-28

Author Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

Maintainer Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

Description This package performs Random Walk with Restart on multiplex and heterogeneous networks. It is described in the following article: ``Random Walk With Restart On Multiplex And Heterogeneous Biological Networks". https://www.biorxiv.org/content/early/2017/08/30/134734 .

License GPL (>= 2)

URL https://www.biorxiv.org/content/early/2017/08/30/134734

Encoding UTF-8

LazyData true

Imports igraph, Matrix, dnet, methods

Depends R(>= 3.5.0)

RoxygenNote 6.0.1

biocViews GenePrediction, NetworkInference, SomaticMutation, BiomedicalInformatics, MathematicalBiology, SystemsBiology, GraphAndNetwork, Pathways, BioCarta, KEGG, Reactome, Network

Suggests BiocStyle, testthat

NeedsCompilation no

git_url https://git.bioconductor.org/packages/RandomWalkRestartMH

git_branch RELEASE_3_13

git_last_commit 0e1c73a

git_last_commit_date 2021-05-19

Date/Publication 2021-10-14

R topics documented:

`compute.adjacency.matrix`

Computes the adjacency matrix of a multiplex network

Description

`compute.adjacency.matrix` is a function to compute the adjacency matrix of a multiplex network provided as a `Multiplex` object.

Usage

```
compute.adjacency.matrix(x,delta = 0.5)
```

Arguments

`x`	A `Multiplex` object describing a multiplex network generated by the function `create.multiplex`.
`delta`	A numeric value between 0 and 1. It sets the probability of performing inter-layer versus intra-layer transitions. It is set by default to 0.5. See more details below.

Details

The parameter `delta` sets the probability to change between layers at the next step. If delta = 0, the particle will always remain in the same layer after a non-restart iteration. On the other hand, if delta = 1, the particle will always change between layers, therefore not following the specific edges of each layer.

Value

A square sparse adjacency matrix created with the `Matrix` package.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

`create.multiplex`,`normalize.multiplex.adjacency`,`compute.transition.matrix`

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
compute.adjacency.matrix(multiObject)
```

---

compute.transition.matrix

Computes the transition matrix of a multiplex and heterogeneous network

---

Description

`compute.transition.matrix` is a function to compute the transition matrix of a multiplex heterogeneous network provided as a `MultiplexHet` object.

Usage

```
compute.transition.matrix(x,lambda = 0.5, delta=0.5)
```

Arguments

`x`	A `MultiplexHet` object describing a multiplex and heterogeneous network generated by the function `create.multiplexHet`.
`lambda`	A numeric value between 0 and 1. It sets the probability of jumping within a network or change to the other network of the heterogeneous system. It is set by default to 0.5. See more details below.
`delta`	A numeric value between 0 and 1. It sets the probability of performing interlayer versus intra-layer transitions. It is set by default to 0.5. See more details below.

Details

We clarify the role of the different parameters in this point:

- `lambda`: For a given node, if a bipartite association exists, the particle can either jump between networks or stay in the current graph with a probability given by this parameter. The closer lambda is to one, the higher is the probability of jumping between networks following bipartite interactions.
- `delta`: This parameter sets the probability to change between layers at the next step. If delta = 0, the particle will always remain in the same layer after a non-restart iteration. On the other hand, if delta = 1, the particle will always change between layers, therefore not following the specific edges of each layer.

Value

A square sparse transition matrix created with the `Matrix` package. It is the transition matrix for the Random Walk with Restart on Multiplex and Heterogeneous networks algorithm.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

`create.multiplexHet`,`compute.adjacency.matrix`

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
h1 <- igraph::graph(c("A","C","B","E","E","D","E","C"), directed = FALSE)
bipartite_relations <- data.frame(m=c(1,3),h=c("A","E"))
multiHetObject <-
    create.multiplexHet(multiObject, h1,bipartite_relations)
compute.transition.matrix(multiHetObject)
```

`create.multiplex` *Create multiplex graphs from individual networks*

Description

`create.multiplex` is a function to create a multiplex network (`Multiplex` object) from up to 6 individual networks defined as `igraph` objects. See more details about multiplex networks below. If just one network is provided, a Multiplex object with one layer is therefore created (A monoplex network).

Usage

```
create.multiplex(...)

## Default S3 method:
create.multiplex(L1, L2 = NULL, L3 = NULL, L4 = NULL,
  L5 = NULL, L6 = NULL, Layers_Name, ...)
```

Arguments

`...`	Further arguments passed to `create.multiplex`
`L1`	An igraph object describing a monoplex network. It will be integrated as the first layer of the multiplex network.
`L2`	An igraph object describing a monoplex network. It will be integrated as the second layer of the multiplex network. It's optional.
`L3`	An igraph object describing a monoplex network. It will be integrated as the third layer of the multiplex network. It's optional.
`L4`	An igraph object describing a monoplex network. It will be integrated as the fourth layer of the multiplex network. It's optional.
`L5`	An igraph object describing a monoplex network. It will be integrated as the fifth layer of the multiplex network. It's optional.
`L6`	An igraph object describing a monoplex network. It will be integrated as the sixth layer of the multiplex network. It's optional.
`Layers_Name`	A vector containing the names of the different layers. This name will be included as an attribute for all the edges of each network. It's optional. See more details below.

Details

A multiplex network is a collection of layers (monoplex networks) sharing the same nodes, but in which the edges represent relationships of different nature. The number of layers of a multiplex object can vary from 1 (monoplex network) up to 6. Therefore, only the first layer is mandatory. We have limited the number of layers to 6 in order to reduce computation times for very large networks.

- `Layers_Name`: A vector contianing the name of the different layers. It's optional, but if provided the number of layers should match the length of this vector. Its elements should be in the same order than the graphs (`Layers_Name = c(L1_name,L2_name,L3_name,L4_name,L5_name,L6_name)`

Value

A Multiplex object. It contains a list of the different graphs integrating the multiplex network, the names and number of its nodes and the number of layers.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

create.multiplexHet,isMultiplex

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
```

---

create.multiplexHet *Create multiplex heterogeneous graphs from individual networks*

Description

create.multiplexHet is a function to create a multiplex and heterogeneous network (MultiplexHet object). It combines a multiplex network composed from 1 (monoplex case) up to 6 layers with another single network whose nodes are of different nature. See more details below.

Usage

```
create.multiplexHet(...)

## Default S3 method:
create.multiplexHet(Multiplex_object, Het_graph,
  Nodes_relations, SecondNetworkName, ...)
```

Arguments

...	Further arguments passed to create.multiplexHet
Multiplex_object	A Multiplex network (Multiplex object) generated by the function create.multiplex. This multiplex network will be integrated as the first network of the heterogeneous network.
Het_graph	An igraph object describing a monoplex network. It will be integrated as the second network of the heterogeneous network.
Nodes_relations	A data frame containing the relationships (bipartite interactions) between the nodes of the multiplex network and the nodes of the second network of the heterogeneous system. The data frame should contain two columns: the first one with the nodes of the multiplex network; the second one with the nodes of the second network. Every node should be present in their corresponing network.
SecondNetworkName	A vector containing the name for the second network to be integrated as part of the heterogeneous network. This name will be included as an attribute for all the edges of this network. It's optional and its default value is SecondNetwork.

Details

A multiplex network is a collection of layers (monoplex networks) sharing the same nodes, but in which the edges represent relationships of different nature. A heterogeneous network is composed of two single networks where the nodes are of different nature. These nodes of different nature are linked through bipartite interactions.

Value

A Multiplex Heterogeneous object. It contains a list of the different graphs integrating the multiplex network, the names and number of its nodes and the number of layers. In addition, it contains the graph of the second network integrating the heterogeneous network along with its number of nodes Finally, it contains a expanded bipartite adjacency matrix describing the relations of the nodes in every layer of the multiplex network with the nodes of the second network.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

create.multiplex, isMultiplexHet

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
h1 <- igraph::graph(c("A","C","B","E","E","D","E","C"), directed = FALSE)
bipartite_relations <- data.frame(m=c(1,3),h=c("A","E"))
create.multiplexHet(multiObject,h1,bipartite_relations)
```

---

create.multiplexHetNetwork.topResults

Creates a Network with the top results of the Random Walk with restart on a Multiplex and Heterogeneous Network

---

Description

`create.multiplexHetNetwork.topResults` is a function to create a network from the top results of the Random Walk with Restart on Multiplex and Heterogeneous networks algorithm (a `RWRMH_Results` object).

Usage

```
create.multiplexHetNetwork.topResults(RWRMH_Results_Object,
    MultiplexHetObject, bipartite_relations, bipartite_name, k=25)
```

Arguments

RWRMH_Results_Object
: A RWRMH_Results object generated by the function Random.Walk.Restart.MultiplexHet representing the results of the Random Ralk with restart on the multiplex and heterogeneous network described in the following argument.

MultiplexHetObject
: A MultiplexHet object generated by the function create.multiplexHet representing a multiplex and heterogeneous network.

bipartite_relations
: A data frame containing the relationships between the nodes of the multiplex network and the nodes of the second network of the heterogeneous system. The data frame should contain two columns: the first one with the nodes of the multiplex network; the second one with the nodes of the second network. Every node should be present in their corresponing network.

bipartite_name
: A vector containing the name for the bipartite relations to be integrated as part of the resulting network. It is included as an attribute for all the bipartite edges of the resulting network. It's optional and its default value is "bipartiteRelations".

k
: A numeric value between 1 and 200. It is the number of top ranked nodes to be included in the resulting multiplex network.

Value

An igraph object containing the top k ranked multiplex nodes and the top k ranked second network nodes in the Random Walk with Restart on a Multiplex and Heterogeneous network algorithm. We include all the possible types of interactions between pairs of nodes according to the different layers of the multiplex network, the bipartite interactions and the second network type of interactions.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

create.multiplexHet, isRWRMH_Results, Random.Walk.Restart.MultiplexHet create.multiplexNetwork.topResu

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
h1 <- igraph::graph(c("A","C","B","E","E","D","E","C"), directed = FALSE)
bipartite_relations <- data.frame(m=c(1,3),h=c("A","E"))
multiHetObject <-
    create.multiplexHet(multiObject, h1,bipartite_relations)
MultiHetTranMatrix <- compute.transition.matrix(multiHetObject)
Multiplex_Seeds <- c(1)
SecondNet_Seeds <- c("E")
RWR_MultiHetResults <-
    Random.Walk.Restart.MultiplexHet(MultiHetTranMatrix, multiHetObject,
```

```
        Multiplex_Seeds,SecondNet_Seeds)
create.multiplexHetNetwork.topResults(RWR_MultiHetResults,multiHetObject,
    bipartite_relations)
```

`create.multiplexNetwork.topResults`
Creates a Network with the top results of the Random Walk with restart on a Multiplex Network

---

Description

`create.multiplexNetwork.topResults` is a function to create a network from the top results of the Random Walk with Restart on Multiplex networks algorithm (a `RWRM_Results` object).

Usage

```
create.multiplexNetwork.topResults(RWRM_Result_Object,
    MultiplexObject,k=25)
```

Arguments

`RWRM_Result_Object`	A `RWRM_Results` object generated by the function `Random.Walk.Restart.Multiplex` representing the results of the Random Ralk with restart on the multiplex network described in the following argument.
`MultiplexObject`	A `Multiplex` object generated by the function `create.multiplex` representing a multiplex network.
`k`	A numeric value between 1 and 200. It is the number of top ranked nodes to be included in the resulting multiplex network.

Value

An `igraph` object containing the top `k` ranked multiplex nodes in the Random Walk with Restart on a Multiplex network algorithm. We include all the possible types of interactions between pairs of nodes according to the different layers of the multiplex network.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

`create.multiplex`, `Random.Walk.Restart.Multiplex` `isRWRM_Results`, `create.multiplexHetNetwork.topResults`

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
AdjMatrix <- compute.adjacency.matrix(multiObject)
AdjMatrixNorm <- normalize.multiplex.adjacency(AdjMatrix)
Seed <- c(1)
RWR_MultiResults <-
    Random.Walk.Restart.Multiplex(AdjMatrixNorm, multiObject, Seed)
create.multiplexNetwork.topResults(RWR_MultiResults,multiObject)
```

Disease_Network *A disease-disease similarity network.*

Description

An igraph object containing a disease-disease similarity network. The network is obtained as described in the article cited in the source section.

Usage

```
data(Disease_Network)
```

Format

An igraph object containing 28246 binary relationships between 6947 diseases.

Source

Valdeolivas, A., Tichit, L., Navarro, C., Perrin, S., Odelin, G., Levy, N., ... Baudot, A. (2017). Random Walk With Restart On Multiplex And Heterogeneous Biological Networks. bioRxiv, 1–31. https://doi.org/10.1101/134734 https://www.biorxiv.org/content/early/2017/08/30/134734

GeneDiseaseRelations *Diseases and their causative genes*

Description

A dataset containing some diseases and their causative genes. The dataset is obtained as described in the article cited in the source section.

Usage

```
data(GeneDiseaseRelations)
```

Format

A data frame with 4496 rows and 2 variables:

hgnc_symbol Gene name, in HGNC format

mim_morbid Disease id, in mim code

Source

Valdeolivas, A., Tichit, L., Navarro, C., Perrin, S., Odelin, G., Levy, N., ... Baudot, A. (2017). Random Walk With Restart On Multiplex And Heterogeneous Biological Networks. bioRxiv, 1–31. https://doi.org/10.1101/134734 https://www.biorxiv.org/content/early/2017/08/30/134734

---

isMultiplex *Is this R object a Multiplex object?*

Description

A Multiplex object is an R object generated as the result of calling the function `create.multiplex` + `isMultiplex(x)` checks whether an R object is Multiplex.

Usage

```
isMultiplex(x)
```

Arguments

x	An R object

Value

A logical constant, TRUE if argument x is a Mutiplex object.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

create.multiplex, isMultiplexHet

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
isMultiplex(multiObject)
isMultiplex(m1)
```

isMultiplexHet *Is this R object a Multiplex Heterogeneous object?*

Description

A Multiplex Heterogeneous object is an R object generated as the result of calling the function `create.multiplexHet`

Usage

```
isMultiplexHet(x)
```

Arguments

x	An R object

Details

`isMultiplexHet(x)` checks whether an R object is MultiplexHet

Value

A logical constant, `TRUE` if argument `x` is a MultiplexHet object.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

`create.multiplexHet`, `isMultiplex`

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
h1 <- igraph::graph(c("A","C","B","E","E","D","E","C"), directed = FALSE)
bipartite_relations <- data.frame(m=c(1,3),h=c("A","E"))
multiHetObject <-
    create.multiplexHet(multiObject,h1,bipartite_relations)
isMultiplexHet(multiHetObject)
isMultiplexHet(h1)
```

`isRWRMH_Results` *Is this R object a RWR on Multiplex-Heterogeneous object (Results of the RWR-MH)?*

Description

A RWR on Multiplex Heterogeneous object is an R object generated as the result of calling the function `Random.Walk.Restart.MultiplexHet` (Results of the RWR-MH)

Usage

```
isRWRMH_Results(x)
```

Arguments

`x`	An R object

Details

`isRWRMH_Results(x)` checks whether an R object is RWRMH_Results

Value

A logical constant, `TRUE` if argument `x` is a RWRMH_Results object.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

`Random.Walk.Restart.MultiplexHet`, `isRWRM_Results`

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
h1 <- igraph::graph(c("A","C","B","E","E","D","E","C"), directed = FALSE)
bipartite_relations <- data.frame(m=c(1,3),h=c("A","E"))
multiHetObject <-
    create.multiplexHet(multiObject,h1,bipartite_relations)
MultiHetTranMatrix <- compute.transition.matrix(multiHetObject)
Multiplex_Seeds <- c(1)
SecondNet_Seeds <- c("E")
RWR_MultiHetResults <-
    Random.Walk.Restart.MultiplexHet(MultiHetTranMatrix,multiHetObject,
    Multiplex_Seeds ,SecondNet_Seeds)
isRWRMH_Results(RWR_MultiHetResults)
```

```
isRWRMH_Results(m1)
```

`isRWRM_Results` *Is this R object a RWR on Multiplex object (Results of the RWR-M)?*

Description

A RWR on Multiplex object is an R object generated as the result of calling the function `Random.Walk.Restart.Multiplex` (Results of the RWR-M)

Usage

```
isRWRM_Results(x)
```

Arguments

x	An R object

Details

`isRWRM_Results(x)` checks whether an R object is RWRM_Results

Value

A logical constant, `TRUE` if argument `x` is a RWRM_Results object.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

`Random.Walk.Restart.Multiplex`, `isRWRMH_Results`

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
AdjMatrix <- compute.adjacency.matrix(multiObject)
AdjMatrixNorm <- normalize.multiplex.adjacency(AdjMatrix)
Seed <- c(1)
RWR_MultiResults <-
    Random.Walk.Restart.Multiplex(AdjMatrixNorm, multiObject,Seed)
isRWRM_Results(RWR_MultiResults)
isRWRM_Results(m1)
```

normalize.multiplex.adjacency
Computes column normalization of an adjacency matrix

Description

normalize.multiplex.adjacency is a function to compute the column normalization of a sparse matrix of the package Matrix.

Usage

```
normalize.multiplex.adjacency(x)
```

Arguments

x	A Matrix object describing an adjacency matrix of a network.

Value

A square sparse column normalized matrix created with the Matrix package.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

compute.adjacency.matrix,Random.Walk.Restart.Multiplex

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
AdjMatrix <- compute.adjacency.matrix(multiObject)
normalize.multiplex.adjacency(AdjMatrix)
```

Pathway_Network *A pathway network (Pathway network)*

Description

An igraph object containing a Pathway network. The network is obtained as described in the article cited in the source section. However, it was reduced in such a way that only contains genes/proteins expressed in the adipose tissue.

Usage

```
data(Pathway_Network)
```

Format

An igraph object containing 62602 binary interactions between 3533 proteins

Source

Valdeolivas, A., Tichit, L., Navarro, C., Perrin, S., Odelin, G., Levy, N., ... Baudot, A. (2017). Random Walk With Restart On Multiplex And Heterogeneous Biological Networks. bioRxiv, 1–31. https://doi.org/10.1101/134734 https://www.biorxiv.org/content/early/2017/08/30/134734

PPI_Network *A protein-protein physical interaction network (PPI network)*

Description

An igraph object containing a protein-protein physical interaction network (PPI network). The network is obtained as described in the article cited in the source section. However, it was reduced in such a way that only contains genes/proteins expressed in the adipose tissue.

Usage

```
data(PPI_Network)
```

Format

An igraph object containing 18062 binary interactions between 4317 proteins

Source

Valdeolivas, A., Tichit, L., Navarro, C., Perrin, S., Odelin, G., Levy, N., ... Baudot, A. (2017). Random Walk With Restart On Multiplex And Heterogeneous Biological Networks. bioRxiv, 1–31. https://doi.org/10.1101/134734 https://www.biorxiv.org/content/early/2017/08/30/134734

Random.Walk.Restart.Multiplex

Performs Random Walk with Restart on a Multiplex Network

Description

Random.Walk.Restart.Multiplex is a function to perform a Random Walk with Restart on a Multiplex network (on a Multiplex object). See more details about the algorithm below.

Usage

```
Random.Walk.Restart.Multiplex(...)

## Default S3 method:
Random.Walk.Restart.Multiplex(x, MultiplexObject, Seeds,
  r = 0.7, tau, ...)
```

Arguments

...	Further arguments passed to Random.Walk.Restart.Multiplex
x	An object of the Matrix package describing a column normalized adjacency matrix of a multiplex network.
MultiplexObject	A Multiplex object generated by the function create.multiplex representing a multiplex network.
Seeds	A vector containing the names of the seeds for the Random Walk algorithm. See more details below.
r	A numeric value between 0 and 1. It sets the probability of restarting to a seed node after each step. See more details below.
tau	A vector containing the probability of restart on the seeds of the different layers (layers weights). It must have the same length than the number of layers of the multpiplex network. The sum of its components divided by the number of layers must be 1. See more details below.

Details

Random Walk with Restart simulates an imaginary particle that starts on a seed(s) node(s) and follows randomly the edges of a network. At each step, there is a restart probability, r, meaning that the particle comes back to the seed(s). The extension to multiplex networks allows the particle to explore different monoplex networks (layers). At each step, the particle can also jump to the same node in a different layer.

- Seeds: A vector containing the name of the different seed node(s). It's mandatory to provide at least one seed. The seed(s) node(s) should belong to any of the layers. The length of this vector should be smaller than the total number of nodes in the multiplex network.

- r: A numeric value representing the restart probability on the seeds for the random walker. It must be between 0 and 1. It is set by default to 0.7, which is the most common value in this kind of approaches. It means that, at each step, the walker has a 70% of probability of coming back to one of the seeds.
- tau: A numeric vector containing the probability of restarting in the nodes of the different layers of the multiplex. In the example below, we define the node 1 as the seed node. However, we can find this node in both layers. Therefore, the walker can restart in any of these seed nodes. It is a way to give different relevance (weight) to the different layers.

Value

A RWRM_Results object. It contains a sorted ranking of all the nodes of the multiplex network, except the seeds, along with their score. In addition, it contains in a different field the nodes used as seeds.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

create.multiplex,compute.adjacency.matrix,normalize.multiplex.adjacency,isRWRM_Results,Random.Walk.

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
AdjMatrix <- compute.adjacency.matrix(multiObject)
AdjMatrixNorm <- normalize.multiplex.adjacency(AdjMatrix)
SeedNodes <- c(1)
Random.Walk.Restart.Multiplex(AdjMatrixNorm,multiObject,SeedNodes)
```

Random.Walk.Restart.MultiplexHet

Performs Random Walk with Restart on a Multiplex and Heterogeneous Network

---

Description

Random.Walk.Restart.MultiplexHet is a function to perform a Random Walk with Restart on a Multiplex and Heterogeneous network (on a MultiplexHet object). See more details about the algorithm below.

Usage

```
Random.Walk.Restart.MultiplexHet(...)

## Default S3 method:
Random.Walk.Restart.MultiplexHet(x, MultiplexHet_Object,
  Multiplex_Seed_Nodes, SecondNet_Seed_Nodes, r = 0.7, tau, eta = 0.5, ...)
```

Arguments

`...`	Further arguments passed to `Random.Walk.Restart.MultiplexHet`
`x`	An object of the `Matrix` package describing the possible transitions in a multiplex and heterogeneous network.
`MultiplexHet_Object`	A `MultiplexHet` object generated by the function `create.multiplexHet` representing a multiplex and heterogeneous network.
`Multiplex_Seed_Nodes`	A vector containing the names of the seeds of the multiplex network for the Random Walk algorithm. See more details below.
`SecondNet_Seed_Nodes`	A vector containing the names of the seeds of the second network for the Random Walk algorithm. See more details below.
`r`	A numeric value between 0 and 1. It sets the probability of restarting to a seed node after each step. See more details below.
`tau`	A vector containing the probability of restart on the seeds of the different multiplex layers (layers weights). It must have the same length than the number of layers of the multpiplex network. The sum of its components divided by the number of layers must be 1. See more details below.
`eta`	A numeric value between 0 and 1. It controls the probability of restarting in each network of the heterogeneous system (Multiplex or second network). See more details below.

Details

Random Walk with Restart simulates an imaginary particle which starts on a seed(s) node(s) and follows randomly the edges of a network. At each step, there is a restart probability, r, meaning that the particle comes back to the seed(s). The extension to multiplex networks allows the particle to explore different monoplex networks (layers). At each step, the particle can also jump to the same node in a different layer. The extension to heterogeneous networks allows the particle to jump between nodes of different nature thanks to bipartite relationships between them. We can combine both, the multiplex and heterogeneous extension, by allowing the particle to jump from a node in every layer of the multiplex network to the other network, and the other way around.

- `Multiplex_Seed_Nodes`: A vector containing the name of the different seed node(s) of the multiplex network. It's mandatory to provide at least one seed (taking in account both types of seeds) The seed(s) node(s) should belong to any of the layers of the multiplex network. The length of this vector should be smaller than the total number of nodes in the multiplex network.

- `SecondNet_Seed_Nodes`: A vector containing the name of the different seed node(s) of the second network. It's mandatory to provide at least one seed (taking in account both types of seeds) The seed(s) node(s) should belong to the second network. The length of this vector should be smaller than the total number of nodes in the second network.
- `r`: A numeric value representing the restart probability on the seeds for the random walker. It must be between 0 and 1. It is set by default to 0.7, which is the most common value in this kind of approaches. It means that, at each step, the walker has a 70% of probability of coming back to one of the seeds.
- `tau`: A numeric vector containing the probability of restarting in the nodes of the different layers of the multiplex. In the example below, we define the node 1 as the seed node. However, we can find this node in both layers. Therefore, the walker can restart in any of these seed nodes. It is a way to give different relevance (weight) to the different layers.
- `eta`: A numeric value between 0 and 1 controlling the probability of restarting in the nodes of each network. In the example below, we define the node 1 as a multiplex seed node and "E" as a second network seed node. Therefore, the walker can restart either in the seed 1 or in the seed "E" with different probabilities (it is a way to give more relevance to the different components of the heterogeneous system). If eta < 0.5 the particle will be more likely to restart in one of the multiplex seeds.

Value

A `RWRMH_Results` object. It contains two sorted rankings: The first one contains the nodes of the multiplex network, except the seeds, along with their score; The second one contains the nodes of the second network, except the seeds, along with their score. In addition, it contains two more fields describing the nodes of different nature used as seeds.

Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

See Also

`create.multiplexHet`, `compute.transition.matrix`, `Random.Walk.Restart.Multiplex` `isRWRMH_Results`

Examples

```
m1 <- igraph::graph(c(1,2,1,3,2,3), directed = FALSE)
m2 <- igraph::graph(c(1,3,2,3,3,4,1,4), directed = FALSE)
multiObject <- create.multiplex(m1,m2)
h1 <- igraph::graph(c("A","C","B","E","E","D","E","C"), directed = FALSE)
bipartite_relations <- data.frame(m=c(1,3),h=c("A","E"))
multiHetObject <-
    create.multiplexHet(multiObject,h1,bipartite_relations)
MultiHetTranMatrix <- compute.transition.matrix(multiHetObject)
Multiplex_Seeds <- c(1)
SecondNet_Seeds <- c("E")
Random.Walk.Restart.MultiplexHet(MultiHetTranMatrix,
    multiHetObject,Multiplex_Seeds,SecondNet_Seeds)
```

Index