

# Package ‘RandomWalkRestartMH’

June 7, 2023

**Type** Package

**Title** Random walk with restart on multiplex and heterogeneous Networks

**Version** 1.21.0

**Date** 2021-08-11

**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://academic.oup.com/bioinformatics/article/35/3/497/5055408>>.

**License** GPL (>= 2)

**URL** <https://github.com/alberto-valdeolivas/RandomWalkRestartMH>

**BugReports** <https://github.com/alberto-valdeolivas/RandomWalkRestartMH/issues>

**Encoding** UTF-8

**LazyData** true

**Depends** R(>= 3.5.0)

**RoxygenNote** 7.1.1

**Imports** igraph, Matrix, dnet, methods

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

**Suggests** BiocStyle, knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**NeedsCompilation** no

**git\_url** <https://git.bioconductor.org/packages/RandomWalkRestartMH>

**git\_branch** devel

**git\_last\_commit** 82b54b5

**git\_last\_commit\_date** 2023-04-25

**Date/Publication** 2023-06-06

**Author** Alberto Valdeolivas [cre, aut, ctb]  
(<https://orcid.org/0000-0001-5482-9023>)  
**Maintainer** Alberto Valdeolivas <alvaldeolivas@gmail.com>

R topics documented:

compute.adjacency.matrix . . . . .	2
compute.transition.matrix . . . . .	3
create.multiplex . . . . .	5
create.multiplexHet . . . . .	6
create.multiplexHetNetwork.topResults . . . . .	7
create.multiplexNetwork.topResults . . . . .	9
Disease_Network . . . . .	10
GeneDiseaseRelations . . . . .	10
isMultiplex . . . . .	11
isMultiplexHet . . . . .	12
isRWRMH_Results . . . . .	13
isRWRM_Results . . . . .	14
normalize.multiplex.adjacency . . . . .	15
Pathway_Network . . . . .	16
PPI_Network . . . . .	16
Random.Walk.Restart.Multiplex . . . . .	17
Random.Walk.Restart.MultiplexHet . . . . .	19
<b>Index</b>	<b>22</b>

---

compute.adjacency.matrix	<i>Computes the adjacency matrix of a multiplex network</i>
--------------------------	---

---

**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(list(m1=m1,m2=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, delta1=0.5, delta2=0.5)
```

**Arguments**

<code>x</code>	A <code>MultiplexHet</code> object describing a multiplex and heterogeneous network generated by the function <code>create.multiplexHet</code> .
<code>lambda</code>	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.

delta1	A numeric value between 0 and 1. It sets the probability of performing inter-layer versus intra-layer transitions in the first multiplex. It is set by default to 0.5. See more details below.
delta2	A numeric value between 0 and 1. It sets the probability of performing inter-layer versus intra-layer transitions in the second multiplex. 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_1 <- create.multiplex(list(m1=m1,m2=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"))
multiObject_2 <- create.multiplex(list(h1=h1))
multiHetObject <- create.multiplexHet(multiObject_1, multiObject_2,
  bipartite_relations)
compute.transition.matrix(multiHetObject)
```

---

create.multiplex	<i>Create multiplex graphs from individual networks</i>
------------------	---

---

## Description

create.multiplex is a function to create a multiplex network (Multiplex object) from a list of 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(LayersList, ...)
```

## Arguments

...	Further arguments passed to create.multiplex
LayersList	A list containing igraph objects describing monoplex networks in every element. We recommend to give names to the different networks (igraph objects).

## 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. At least a list with one element, an igraph object, should be provided.

## 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(list(m1=m1,m2=m2))
```

---

<code>create.multiplexHet</code>	<i>Create multiplex heterogeneous graphs from individual networks</i>
----------------------------------	---

---

## 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_1,
  Multiplex_object_2,
  Nodes_relations,
  ...
)
```

## Arguments

<code>...</code>	Further arguments passed to <code>create.multiplexHet</code>
<code>Multiplex_object_1</code>	First Multiplex network ( <code>Multiplex</code> object) generated by the function <code>create.multiplex</code> . This multiplex network will be integrated as the first network of the heterogeneous network.
<code>Multiplex_object_2</code>	Second Multiplex network ( <code>Multiplex</code> object) generated by the function <code>create.multiplex</code> . This multiplex network will be integrated as the first network of the heterogeneous network.
<code>Nodes_relations</code>	A data frame containing the relationships (bipartite interactions) between the nodes of the first multiplex network and the nodes of the second multiplex of the heterogeneous system. The data frame should contain two or three columns: the first one with the nodes of the multiplex network; the second one with the nodes of the second network. The third one is not mandatory and it should contain the weights. Every node should be present in their corresponding multiplex network.

## 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 an 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_1 <- create.multiplex(list(m1=m1,m2=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"))
multiObject_2 <- create.multiplex(list(h1=h1))
create.multiplexHet(multiObject_1, multiObject_2,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 Walk 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 (bipartite interactions) between the nodes of the first multiplex network and the nodes of the second multiplex of the heterogeneous system. The data frame should contain two or three columns: the first one with the nodes of the multiplex network; the second one with the nodes of the second network. The third one is not mandatory and it should contain the weights. Every node should be present in their corresponding multiplex 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.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_1 <- create.multiplex(list(m1=m1,m2=m2))
h1 <- igraph::graph(c("A","C","B","E","E","D","E","C"), directed = FALSE)
multiObject_2 <- create.multiplex(list(h1=h1))
bipartite_relations <- data.frame(m=c(1,3),h=c("A","E"))
multiHetObject <-
  create.multiplexHet(multiObject_1, multiObject_2,bipartite_relations)
MultiHetTranMatrix <- compute.transition.matrix(multiHetObject)
Multiplex1_Seeds <- c(1)
Multiplex2_Seeds <- c("E")
RWR_MultiHetResults <-
  Random.Walk.Restart.MultiplexHet(MultiHetTranMatrix, multiHetObject,
    Multiplex1_Seeds,Multiplex2_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 Walk 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(list(m1=m1,m2=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	<i>A disease-disease similarity network.</i>
-----------------	--

---

### 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	<i>Diseases and their causative genes</i>
----------------------	---

---

### 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](mailto: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(list(m1=m1,m2=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_1 <- create.multiplex(list(m1=m1,m2=m2))
h1 <- igraph::graph(c("A","C","B","E","E","D","E","C"), directed = FALSE)
multiObject_2 <- create.multiplex(list(h1=h1))
bipartite_relations <- data.frame(m=c(1,3),h=c("A","E"))
multiHetObject <-
  create.multiplexHet(multiObject_1,multiObject_2,bipartite_relations)
isMultiplexHet(multiHetObject)
isMultiplexHet(h1)
```

---

isRWRMH_Results	<i>Is this R object a RWR on Multiplex-Heterogeneous object (Results of the RWR-MH)?</i>
-----------------	--

---

## 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](#), [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_1 <- create.multiplex(list(m1=m1,m2=m2))
h1 <- igraph::graph(c("A","C","B","E","E","D","E","C"), directed = FALSE)
multiObject_2 <- create.multiplex(list(h1=h1))
bipartite_relations <- data.frame(m=c(1,3),h=c("A","E"))
multiHetObject <-
  create.multiplexHet(multiObject_1,multiObject_2,bipartite_relations)
MultiHetTranMatrix <- compute.transition.matrix(multiHetObject)
Multiplex1_Seeds <- c(1)
Multiplex2_Seeds <- c("E")
RWR_MultiHetResults <-
  Random.Walk.Restart.MultiplexHet(MultiHetTranMatrix,multiHetObject,
    Multiplex1_Seeds ,Multiplex2_Seeds)
```

```
isRWRMH_Results(RWR_MultiHetResults)
isRWRMH_Results(m1)
```

---

isRWRM_Results	<i>Is this R object a RWR on Multiplex object (Results of the RWR-M)?</i>
----------------	---

---

## 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(list(m1=m1,m2=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(list(m1=m1,m2=m2))
AdjMatrix <- compute.adjacency.matrix(multiObject)
normalize.multiplex.adjacency(AdjMatrix)
```

---

Pathway_Network	<i>A pathway network (Pathway network)</i>
-----------------	--

---

**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	<i>A protein-protein physical interaction network (PPI network)</i>
-------------	---

---

**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,
  MeanType = "Geometric",
  DispResults = "TopScores",
  ...
)
```

## 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 multiplex network. The sum of its components divided by the number of layers must be 1. See more details below.
MeanType	The user can choose one of the following options: c("Geometric", "Arithmetic", "Sum"). These options represent the different way to combine the RWR score for the same node in different layers. By default and recommended Geometric (Geometric Mean.). Arithmetic is the arithmetic mean and sum just sum all the scores for the same node across the different layers.

**DispResults**      The user can choose one of the following options: `c("TopScores","Alphabetic")`. These options represent the way the RWR results would be presented. By default, and recommended, the nodes would be ordered by score. This option is also required to properly run the `create.multiplexNetwork.topResults` and `create.multiplexHetNetwork.topResults` functions

## 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.Restart.MultiplexHet](#)

## 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(list(m1=m1,m2=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,
  Multiplex1_Seeds,
  Multiplex2_Seeds,
  r = 0.7,
  tau1,
  tau2,
  eta = 0.5,
  MeanType = "Geometric",
  DispResults = "TopScores",
  ...
)
```

## 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.
Multiplex1_Seeds	A vector containing the names of the seeds of the first multiplex network for the Random Walk algorithm. See more details below.
Multiplex2_Seeds	A vector containing the names of the seeds of the second multiplex network for the Random Walk algorithm. See more details below.

<code>r</code>	A numeric value between 0 and 1. It sets the probability of restarting to a seed node after each step. See more details below.
<code>tau1</code>	A vector containing the probability of restart on the seeds of the different multiplex layers (layers weights) for the first multiplex. It must have the same length than the number of layers of the multiplex. network. The sum of its components divided by the number of layers must be 1. See more details below.
<code>tau2</code>	A vector containing the probability of restart on the seeds of the different multiplex layers (layers weights) for the second multiplex. It must have the same length than the number of layers of the multiplex. network. The sum of its components divided by the number of layers must be 1. See more details below.
<code>eta</code>	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.
<code>MeanType</code>	The user can choose one of the following options: <code>c("Geometric","Arithmetic","Sum")</code> . These options represent the different way to combine the RWR score for the same node in different layers. By default and recommended Geometric (Geometric Mean.). Arithmetic is the arithmetic mean and sum just sum all the scores for the same node across the different layers.
<code>DispResults</code>	The user can choose one of the following options: <code>c("TopScores","Alphabetic")</code> . These options represent the way the RWR results would be presented. By default, and recommended, the nodes would be ordered by score. This option is also required to properly run the

## 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 three sorted rankings: i) The first one contains the global results, i.e. the nodes of both multiplex networks along with their score; ii) The second one contains the nodes of the first multiplex network, except the seeds, along with their score. iii) The last one contains the nodes of the second multiplex network, excepting the seeds, along with their score. In addition, it contains one more field describing the nodes used as seeds.

### Author(s)

Alberto Valdeolivas Urbelz <alvaldeolivas@gmail.com>

### See Also

`create.multiplexHet`, `compute.transition.matrix`, `Random.Walk.Restart.Multiplex` is RWRMH\_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_1 <- create.multiplex(list(m1=m1,m2=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"))
multiObject_2 <- create.multiplex(list(h1=h1))
multiHetObject <- create.multiplexHet(multiObject_1, multiObject_2,
  bipartite_relations)
MultiHetTranMatrix <- compute.transition.matrix(multiHetObject)
Multiplex1_Seeds <- c(1)
Multiplex2_Seeds <- c("E")
Random.Walk.Restart.MultiplexHet(MultiHetTranMatrix,
  multiHetObject,Multiplex1_Seeds,Multiplex2_Seeds)
```

# Index

## \* datasets

Disease\_Network, [10](#)  
GeneDiseaseRelations, [10](#)  
Pathway\_Network, [16](#)  
PPI\_Network, [16](#)

compute.adjacency.matrix, [2](#), [4](#), [15](#), [18](#)  
compute.transition.matrix, [3](#), [3](#), [21](#)  
create.multiplex, [3](#), [5](#), [7](#), [9](#), [11](#), [18](#)  
create.multiplexHet, [4](#), [5](#), [6](#), [8](#), [12](#), [21](#)  
create.multiplexHetNetwork.topResults,  
[7](#), [9](#)  
create.multiplexNetwork.topResults, [8](#),  
[9](#)

Disease\_Network, [10](#)

GeneDiseaseRelations, [10](#)

isMultiplex, [5](#), [11](#), [12](#)  
isMultiplexHet, [7](#), [11](#), [12](#)  
isRWRM\_Results, [9](#), [13](#), [14](#), [18](#)  
isRWRMH\_Results, [8](#), [13](#), [14](#), [21](#)

normalize.multiplex.adjacency, [3](#), [15](#), [18](#)

Pathway\_Network, [16](#)  
PPI\_Network, [16](#)

Random.Walk.Restart.Multiplex, [9](#), [14](#), [15](#),  
[17](#), [21](#)  
Random.Walk.Restart.MultiplexHet, [8](#), [13](#),  
[18](#), [19](#)