mully - an R Package to create, modify and visualize multilayered graphs

true

2021-10-14

Introduction

Network theory has been used for many years in the modeling and analysis of complex systems, as epidemiology, biology and biomedicine . As the data evolves and becomes more heterogeneous and complex, monoplex networks become an oversimplification of the corresponding systems. This imposes a need to go beyond traditional networks into a richer framework capable of hosting objects and relations of different scales, called Multilayered Network **Mully**, **mul**ti**la**y**er** networks, is an R package that provides a multilayer network framework. Using this package, the user can create, modify and visualize graphs with multiple layers. This package is an extension to the igraph package that provides a monolayer graph framework. The package is implemented as a part of the Multipath Project directed by Dr. Frank Kramer . ## Publication More information and references can be found in the mully paper:

https://www.mdpi.com/2073-4425/9/11/519

Installation

Installation via Github

```
require(devtools)
install_github("frankkramer-lab/mully")
```

Load the package

```
library("mully")
#>
#> Attaching package: 'mully'
#> The following object is masked from 'package:utils':
#>
#>     demo
#> The following object is masked from 'package:base':
#>
#>     merge
```

Test the package

In this section, we provide a demo to test the package by calling some of the function. After running this script, you will have a graph g with 3 layers and 8 nodes. the graph can also be modified by calling other functions. Please refer to help to see the available functions. ### Create new mully graph

```r
g=mully("MyFirstMully",direct = F)
```

Add Layers

```r
g=addLayer(g, c("Gene", "Drug", "Disease"))
```

Add/print Nodes

```r
g=addNode(g,"d1","disease",attributes=list(type="t1"))

g=addNode(g,"d2","disease",attributes=list(type="t1"))

g=addNode(g,"d3","disease",attributes=list(type="t1"))

g=addNode(g,"dr1","drug",attributes=list(effect="strong"))

g=addNode(g,"dr2","drug",attributes=list(effect="strong"))

g=addNode(g,"dr3","drug",attributes=list(effect="moderate"))

g=addNode(g,"g1","gene",attributes=list(desc="AF"))

g=addNode(g,"g2","gene",attributes=list(desc="BE"))

#See vertices attributes
print(getNodeAttributes(g))
```

Add/print/remove Edges

```r
g=addEdge(g,"dr1","d2",list(name="treats"))
g=addEdge(g,"dr1","d2",list(name="extraEdge"))
g=addEdge(g,"d2","g1",list(name="targets"))
g=addEdge(g,"g2","dr3",list(name="mutates and causes"))
g=addEdge(g,"dr3","d3",list(name="treats"))

print(getEdgeAttributes(g))

removeEdge(g,"d2","dr1",multi=T)
```

Merge two graphs

```r
#Create a Second graph
g1=mully()

g1=addLayer(g1,c("protein","drug","gene"))

g1=addNode(g1,"dr4","drug",attributes=list(effect="strong"))
g1=addNode(g1,"dr5","drug",attributes=list(effect="strong"))
g1=addNode(g1,"dr6","drug",attributes=list(effect="moderate"))

g1=addNode(g1,"p1","protein")
g1=addNode(g1,"p2","protein")
g1=addNode(g1,"p3","protein")

g1=addNode(g1,"g3","gene")
g1=addNode(g1,"g4","gene")

g1=addEdge(g1,nodeStart = "p2",nodeDest = "p3",attributes = list(name="interacts"))
g1=addEdge(g1,nodeStart = "dr6",nodeDest = "g4",attributes = list(name="targets"))

#Merge both graphs
g12=merge(g,g1)

#Print the graph
print(g12)
```

Visualization

```r
plot(g12,layout = "scaled")
```

```r
plot3d(g12)
```

Available Functions

mully functions are divided into different files depending on their functionnality range: Constructor , Layers Functions , Node Functions , Edge Functions , Merge Function , Visualization Functions , Import Functions , Export Functions , Demo.

Function	Description
`mully(name,direct)`	Constructor Function, Create an empty multilayered graph
`print(g)`	Print function
`addLayer(g, nameLayer)`	Add a layer or a set of layers to a graph
`removeLayer(g, name,trans)`	Delete a layer or a set of layers from a graph
`isLayer(g, name)`	Verify if the layer exists in a graph
`getLayersCount(g)`	Get the number of layers in a graph
`getLayer(g, nameLayer)`	Get the nodes on a layer in a graph

Function	Description
`getNode(g,nameNode)`	Get a node from a graph
`getIDNode(g,nameNode)`	Get the id of a node
`addNode(g, nodeName, layerName, attributes)`	Add a node with assigned layer and attributes to a graph
`removeNode(g, name,trans)`	Delete a node or a set of nodes from a graph
`getNodeAttributes(g,nameNode)`	Get the attributes of one or all nodes
`addEdge(g, nodeStart, nodeDest, attributes)`	Add an edge
`removeEdge(g, nodeStart, nodeDest,attributes, multi)`	Delete an edge
`getEdgeAttributes(g,nodeStart,nodeDest)`	Get the attributes of the edges connecting two nodes or all the edges in the graph
`getIDEdge(g,nodeStart,nodeDest)`	Get the ids of the edges connecting two nodes
`merge(g1,g2)`	Merge or unite two graphs
`plot(g,layout)`	Plot the graph in 2D
`plot3d(g)`	Plot the graph in 3D using rgl
`importGraphCSV(name,direct,layers,nodes,edges)`	Import a mully graph from csv files
`importLayersCSV(g,file)`	Import layers to a mully graph from a CSV file
`importNodesCSV(g,file)`	Import nodes to a mully graph from a CSV file
`importEdgesCSV(g,file)`	Import edges to a mully graph from a CSV file

addEdge

Add an edge

The function is used to add an edge to a given mully graph. mully supports multi-edges, i.e. two nodes can be connected with multiple edges. The uniqueness of the edges is based on the connection with the attributes. When adding a new connection between two nodes that are already cnnected, the new edge should have different attributes. The function needs the following arguments:

- **g** - The input graph
- **nodeStart** - The first endpoint of the edge
- **nodeDest** - The second endpoint of the edge
- **attributes** - A named list, the attributes to assign to the edge

The function returns the graph, with the added edge.

Example

```r
g=mully::demo()
addEdge(g,"dr3","g2",attributes=list(name="newEdge"))
```

addLayer

Add a layer or a set of layers to a graph

The function is used to add a new layer to a given mully graph. The layer name should be unique and should not already exist in the graph. The names are not case-sensitive. the function has an internal count, and assigns new IDs to new layers. The internal counr number only increases, i.e. it does not change after deleting a lyer. Therefore the ID of the last added layer is not necessarily the number of layers in the mully graph. The function needs the following arguments:

- **g** - The input graph
- **nameLayer** - The name or the list of the names of the layers to be added. The layer names must be unique

The function returns the graph, with the layers added.

Example

```
g=mully::demo()
addLayer(g,"Complex")
```

addNode

Add a node with assigned layer and attributes to a graph

The function is used to add a node to a given mully graph. The layer to which this node should be added is required. Nodes on single layers should be unique, but can not be on different layers. Layer Names are not case-sensitive. The function needs the following arguments:

- **g** - The input graph
- **nodeName** - The name of the node to add
- **layerName** - The name of the layer to be assigned to the node
- **attributes** - The attributes of the node to add. This argument must be a named list

The function returns the graph, with the new node.

Example

```
g=mully::demo()
attributes=list("specie"="Homo Sapiens")
addNode(g = g,nodeName = "g3",layerName = "Gene",attributes = attributes)
```

exportCSV

Export mully into CSV files

The function is used to export a given mully graph in the CSV Format. Three files will be generated, respectively containing the layers', nodes' and edges' information. The function needs the following arguments:

- **g** - The input graph
- **target** - The target file in which the files will be generated. By default the WD.

Example

```
g=mully::demo()
exportCSV(g)
```

getEdgeAttributes

Get the attributes of the edges connecting two nodes

The function is used to obtain information on the edges in a given mully graph. The information can be on a single edge or all the edges in the graph. The function needs the following arguments:

- **g -** The input graph
- **nodeStart -** The first endpoint of the edge
- **nodeDest -** The second endpoint of the edge

The function returns a dataframe containing the edges with their attributes. If both nodes' arguments are missing, it returns the complete information on edges and their attributes.

Example

```
g=mully::demo()
##Print all Edges
getEdgeAttributes(g)
##Get a Single Edge
getEdgeAttributes(g,"d2","g1")
```

getIDEdge

Get the ids of the edges connecting two nodes

The function is used to get the internal ID of an edge in a given mully graph. The function needs the following arguments:

- **g -** The input graph
- **nodeStart -** The first endpoint of the edge
- **nodeDest -** The second endpoint of the edge

The nodes passed to this function as arguments should be the unique names assigned to the nodes upon addition. mully supports multi-edges, therefore this function may return a list of edges connecting two nodes. The function returns a list containing the ids of the edges connecting the nodes.

Example

```
g=mully::demo()
getIDEdge(g,"d2","dr1")
```

getIDNode

Get the id of a node

The function is used to get the internal ID of a given node in a given mully graph. The function needs the following arguments:

- **g -** The input graph
- **nameNode -** The name of the node

The function returns the id of the specified node.

getLayer

Get the nodes on a layer in a graph

The function is used to get the nodes on a given layer in a given mully graph. The layer name is not case-sensitive. The function needs the following arguments:

- **g -** The input graph
- **nameLayer -** The name of the layer

The function returns a List of the nodes on the given layer.

Example

```
g=mully::demo()
getLayer(g,"gene")
```

getLayersCount

Get the number of layers in a graph

The function is used to get the number of layers in a given mully graph. The function needs the following arguments:

- **g -** The input graph

The function returns the count of the layers.

Example

```
g=mully::demo()
getLayersCount(g)
```

getNode

Get a node from a graph

The function is used to get a node from a given mully graph as an igraph.vs object. The function needs the following arguments:

- **g -** The input graph
- **name -** The name of the node

The function returns the node as igraph.vs.

getNodeAttributes

Get the attributes of a node

The function is used to get a node or a list of nodes with corresponding attributes from a given mully graph. If the node name is missing, the complete node information is extracted. The function needs the following arguments:

- **g -** The input graph
- **nameNode -** The name of the node
- **layerByName -** A boolean to specify whether to export the layers by name or by ID

The function returns a dataframe containing the attributes of the specified node.

Example

```
g=mully::demo()
getNodeAttributes(g,layerByName = TRUE)
```

importEdgesCSV

Import Edges to a mully graph from a CSV file

The function is used to import edges to a mully graph. The function reads a CSV file and adds edges between existing nodes to the graph. The graph should already contain layers and nodes. The function needs the following arguments:

- **g -** The mully graph to which the nodes will be added. The graph should already have the layers and the nodes.
- **file -** The path to the CSV file containing the edges' information

The function returns the mully graph with the added edges.

importGraphCSV

Import a mully graph from CSV files

The function is used to create a graph from CSV files. To create the graph, three files are needed: the layers, nodes and edges. mully also offers functions to import individual files containing nodes', edges' and layers information. See Functions `importLayersCSV()`, `importNodesCSV()` and `importEdgesCSV()`. The function needs the following arguments:

- **name -** The name of the graph
- **direct -** A boolean to indicate if the graph is directed or not
- **layers -** The path to the CSV file containing the layers' information
- **nodes -** The path to the CSV file containing the nodes' information
- **edges -** The path to the CSV file containing the edges' information

The function returns a new mully graph.

importLayersCSV

Import Layers to a mully graph from a CSV file

The function is used to add layers to a mully graph. The function reads a CSV file and adds the layers to the graph. The file should contain the layers' names. Layer IDs are assigned automatically. The function needs the following arguments:

- **g -** The mully graph to which the layers will be added. If missing, a new mully graph is created
- **file -** The path to the CSV file containing the layers' information

The function returns the mully graph with the added layers.

importNodesCSV

Import Nodes to a mully graph from a CSV file

The function is used to import nodes into an existing mully graph. The function reads a CSV file and adds the nodes with the attributes to the graph. The graph should already contain layers, and the nodes in the file should have an attribute n referring to the layer assignment. The function needs the following arguments:

- **g** - The mully graph to which the nodes will be added. The graph should already have the layers
- **file** - The path to the CSV file containing the nodes' information
- **name** - The name of the column containing the names of the nodes

The function returns the mully graph with the added nodes.

is.mully

Is this a mully graph?

The function check if a given graph is a mully graph. The function needs the following arguments:

- **g** - The input graph

The function returns a boolean indicating whether the graph is or not a mully object. ### isLayer **Verify if the layer exists in a graph**

This function is used to check if a given mully graph contains a given layer. The layer name is not case-sensitive. The function needs the following arguments:

- **g** - The input graph
- **name** - The name of the layer

The function returns a boolean value.

Example

```
g=mully::demo()
isLayer(g,"Gene")
isLayer(g,"Complex")
```

merge

Merge or unite two graphs

The function is used to merge two mully graphs. The merge is layer based, i.e. nodes belonging to similar layers in both graphs will be combined in the returned graph. The node names are unique on single layers but can be redundant over different layers. The merge is based on the first arguments, therefore all elements of the second argument will be added to the first. The function needs the following arguments:

- **g1** - The first graph to merge. This is the base of the merge.
- **g2** - The second graph to merge. All of its elements are added to the first graph.

The function returns the merge of the two graphs. The merge is based on the first given graph.

Example

```r
#Create First Graph
g=mully::demo()

#Create Second Graph
g1 <- mully("MySecondMully",direct = F)

g1 <- addLayer(g1, c("gene", "Protein", "Drug"))

g1=addNode(g1,"p1","Protein",attributes=list(type="t1"))

g1=addNode(g1,"p2","Protein",attributes=list(type="t1"))

g1=addNode(g1,"p3","Protein",attributes=list(type="t1"))

g1=addNode(g1,"dr6","drug",attributes=list(effect="strong"))

g1=addNode(g1,"dr7","drug",attributes=list(effect="strong"))

g1=addNode(g1,"dr8","drug",attributes=list(effect="moderate"))

g1=addNode(g1,"g3","gene",attributes=list(desc="AF"))

g1=addNode(g1,"g9","gene",attributes=list(desc="BE"))

g1=addEdge(g1,"dr8","g9",list(name="targets"))

g1=addEdge(g1,"p3","p2",list(name="interactWith"))

#Merge Graphs
merge(g,g1)
```

mully

Create an empty multilayered graph

The function create an empty mully graph. The function needs the following arguments:

- **name** - The name to be assigned to the graph.
- **direct** - A boolean value, if the graph is directed or not. By default TRUE.

The function returns the created multilayered graph. ### plot,mully **Plot the graph in 2D**

The function is used to plot the mully graph. It uses the plot function from igraph. We do not recommend using this function to plot big graphs, and use plot3d instead. The function needs the following arguments:

- **g** The input graph
- **layout** The layout. Can either be random or scaled

plot3d

Plot the graph in 3D using rgl

The funstion is used to generate 3D interactive plots of a mully graph. The 3D plot is generated using the R Package rgl, and uses some of the arguments passed to the igraph function rgl.plot. Most of the arguments are set to default values, but can be changed. The layout is calculated internally based on the layers. The function needs the following arguments:

- **g** The input graph
- **layers** A boolean whether to add the layers or not
- **vertex.label** The vertices' labels
- **vertex.label.color** The vertices' colors. If not specified, the colors will be chosen randomly
- **vertex.plac** The placement form of the vertices on the layer. Can either be "circle" which will place them on a circle, or "disc" which will place them randomly on a disc. The default is "circle"
- **edge.color** The edges' colors.If not specified, inter-edges are black, and intra-edges have the same color as the nodes on the layer
- **edge.width** The edge width. Default set to 5.
- **edge.arrow.size** The edges' arrow size. Default set to 10
- **edge.arrow.width** The edges' arrow width. Default set to 1

The function can take the following arguments supported and not ignored by rglplot}: vertex.label, vertex.label.color, edge.color, edge.width, edge.arrow.size,edge.arrow.width.

print,mully

Print a mully graph

The function prints a mully graph. The function needs the following arguments:

- **g -** The input graph

Example

```r
g=mully::demo()
print(g)
```

removeEdge

Delete an edge

The function is used to remove an edge from a given mully graph. Since mully supports multi-edges, a boolean *multi* is needed to specify whether to delete multiple edges or a single edge. In case one edge should be deleted, the named list of attributes of the edge is required. The function needs the following arguments:

- **g -** The input graph
- **nodeStart -** The first endpoint of the edge
- **nodeDest -** The second endpoint of the edge
- **attributes -** The attributes of the edge to delete. Required if the nodes are multi-connected
- **multi -** A boolean. Specifies whether to delete multiple edges or not, in case they exist.

The function returns the graph with the deleted edges.

Example

```
g=mully::demo()
removeEdge(g,"dr1","d2",multi=TRUE)
```

removeLayer

Delete a layer or a set of layers from a graph

The function is used to remove a layer from a given mully graph. Removing a layer results deleting all nodes assigned to the layer, and the edges connecting these nodes to other nodes in the graph. Transitive edges between nodes can be added by setting trans to TRUE. The function needs the following arguments:

- **g** - The input graph
- **name** - The name or the list of the names of the layers to be deleted
- **trans** - A boolean whether to insert transitive edges or not

The function returns the graph, with the given layer and its corresponding nodes and edges removed.

Example

```
g=mully::demo()
removeLayer(g,"gene",trans=TRUE)
```

removeNode

Delete a node or a set of nodes from a graph

The function is used to remove a node from a given mully graph. Removing a node results removing the edges connecting this node to other nodes in the graph. Transitive edges can also be added after deletion by setting trans to TRUE (a->b, b->c, removing b with trans=TRUE will add a->c). The function needs the following arguments:

- **g** - The input graph
- **name** - The name or the list of names of the nodes to be deleted
- **trans** - A boolean whether to insert transitive edges or not

The function returns the graph, with the nodes deleted.

Example

```
g=mully::demo()
removeNode(g,"dr1",trans=TRUE)
```