DynComm Package Manual

April 7, 2019

`DynComm$addRemoveEdgesFile`
addRemoveEdgesFile(graphAddRemoveFile)

Description

This method reads edges from a file and adds or removes them from the graph.

The file must have only one edge per line, with values separated by a white space (both SPACE and TAB work in any amount and combination).

The first value is the source node, the second is the destination node, and the third is the weight.

The weight can be ommited if the edge is to be added using the default weight of 1 (one), or if the parameter to ignore weights was set.

If the weight is exactly zero, the edge is removed from the graph.

If a node, mentioned in the source or destination, does not exist it will be added to the graph.

The method detects automatically if the weight is present on a row by row basis so some rows may have weights defined and others not.

Usage

```
addRemoveEdgesFile(graphAddRemoveFile)
```

Value

FALSE if any kind of error occurred. Otherwise, TRUE

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$addRemoveEdgesFile("graphAddRemoveFile.txt")
```

ALGORITHM *List of available algorithms.*

Description

An algorithm mainly defines how nodes and/or communities are processed, when quality measurements occur and what happens to the communities depending on the value of the quality obtained.

Usage

```
ALGORITHM$algorithm
```

Format

A named list with the names of the available algorithms:

algorithm See available algorithms below.

Currently supported algorithms

Louvain is a greedy optimization method to extract communities from large networks by optimizing the density of edges inside communities to edges outside communities.

Examples

```
ALGORITHM$LOUVAIN
```

DynComm$communities *communities()*

Description

This method returns all communities after the last iteration.

Usage

```
communities()
```

Value

a list of all communities

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$communities()
```

DynComm$communityCount

communityCount()

Description

Get the number of communities after the last iteration of the algorithm.

Usage

```
communityCount()
```

Value

an unsigned integer value with the number of communities

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$communityCount()
```

DynComm$communityEdgeWeight

communityEdgeWeight(source,destination)

Description

Get the weight of the edge that goes from source to destination after the last iteration.

Usage

```
communityEdgeWeight(source,destination)
```

Arguments

source	The name of the source node that is part of the edge
destination	The name of the destination node that is part of the edge

Value

a floating point number with the weight

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$communityEdgeWeight(12,42)
```

DynComm$communityInnerEdgesWeight
communityInnerEdgesWeight(community)

Description

Get the sum of weights of the inner edges of the given community after the last iteration.

Usage

```
communityInnerEdgesWeight(community)
```

Arguments

community	The name of the intended community

Value

a floating point number with the weight

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$communityInnerEdgesWeight(1)
```

DynComm$communityMapping
communityMapping()

Description

Get the community mapping for all communities after the last iteration.

Usage

```
communityMapping()
```

Value

a two column matrix with communities in the first column and the nodes in the second

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$communityMapping()
```

DynComm$communityNodeCount *communityNodeCount(community)*

Description

Get the amount of nodes in the given community after the last iteration.

Usage

```
communityNodeCount(community)
```

Arguments

community	The name of the intended community

Value

an unsigned integer with the number nodes in the given community

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$communityNodeCount(3)
```

DynComm$community *community(node)*

Description

Get the community of the given node after the last iteration.

Usage

```
community(node)
```

Arguments

node	The name of the intended node

Value

an unsigned integer with the community of the given node

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$community(8)
```

DynComm$communityTotalWeight *communityTotalWeight(community)*

Description

Get the sum of weights of all edges of the given community after the last iteration.

Usage

```
communityTotalWeight(community)
```

Arguments

community	The name of the intended community

Value

a floating point number with the weight

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$communityTotalWeight(1)
```

---

densopt *The implementation of eTILES algorithm.*

Description

The implementation of eTILES algorithm.

Usage

```
densopt(graph, graph.directed = TRUE, comms = NULL,
  type.names = c("num", "alfa"))
```

Arguments

graph	input edge list graph
graph.directed	is graph directed or not
comms	community node assignment for each node in graph
type.names	are node names numeric or alfanumeric

Value

dataframe with nodes and new assigned community

See Also

nchar which this function wraps

Examples

```
str_length(letters)
```

DynComm-package *DynComm: Dynamic Network Communities Detection*

Description

Bundle of algorithms used for evolving network analysis regarding community detection. Implements several algorithms, using a common API, that calculate communities for graphs whose nodes and edges change over time. Edges, which can have new nodes, can be added or deleted, and changes in the communities are calculated without recalculating communities for the entire graph.

DynComm *DynComm*

Description

This class provides a single interface for all algorithms in the different languages. It provides methods to get results of processing and to interact with the nodes, edges and communities.

Usage

```
DynComm(algorithm, quality, parameters)
```

Arguments

parameters	A two column matrix defining additional parameters. See the Parameters section on this page
algorithm	One of the available algorithms. See ALGORITHM
quality	One of the available quality measurement functions. See QUALITY

Value

`DynComm` object

Parameters

A two column matrix defining additional parameters to be passed to the selected algorithm and quality measurement function. The first column names the parameter and the second defines its value.

-c Owsinski-Zadrozny quality function parameter. Values [0.0:1.0]. Default: 0.5

-k Shi-Malik quality function kappa_min value. Value > 0 . Default 1

-w Treat graph as weighted. In other words, do not ignore weights for edges that define them when inserting edges in the graph. A weight of exactly zero removes the edge instead of inserting so its weight is never ignored. Without this parameter defined or for edges that do not have a weight defined, edges are assigned the default value of 1 (one). As an example, reading from a file may define weights (a third column) for some edges (defined in rows, one per row) and not for others. With this parameter defined, the edges that have weights that are not exactly zero, have their weight replaced by the default value.

-e Stops when, on a cycle of the algorithm, the quality is increased by less than the value given in this parameter.

Methods

getAlgorithm() Get the algorithm being used. See DynComm$getAlgorithm

addRemoveEdgesFile(graphAddRemoveFile) Add and remove edges read from a file. See DynComm$addRemoveEdgesFile

quality() Get the quality measurement of the graph after the last iteration. See DynComm$quality

communityCount() Get the number of communities after the last iteration. See DynComm$communityCount

communities() Get all communities after the last iteration. See DynComm$communities

communityInnerEdgesWeight(community) Get the sum of weights of the inner edges of the given community after the last iteration. See DynComm$communityInnerEdgesWeight

communityTotalWeight(community) Get the sum of weights of all edges of the given community after the last iteration. See DynComm$communityTotalWeight

communityEdgeWeight(source,destination) Get the weight of the edge that goes from source to destination after the last iteration. See DynComm$communityEdgeWeight

communityNodeCount(community) Get the amount of nodes in the given community after the last iteration. See DynComm$communityNodeCount

community(node) Get the community of the given node after the last iteration. See DynComm$community

nodesCount() Get the total number of nodes after the last iteration. See DynComm$nodesCount

nodesAll() Get all nodes in the graph after the last iteration. See DynComm$nodesAll

nodes(community) Get all nodes belonging to the given community after the last iteration. See DynComm$nodes

communityMapping() Get the community mapping for all communities after the last iteration.See DynComm$communityMapping

time() Get the cumulative time spent on processing after the last iteration. See DynComm$time

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$addRemoveEdgesFile("initial_graph.txt")
dc$communityCount()
dc$communities()
dc$communityNodeCount(1)
dc$nodes(1)
dc$communityMapping(TRUE)
dc$time()
dc$addRemoveEdgesFile("s0000000000.txt")
```

`etiles` *The implementation of eTILES algorithm.*

Description

The implementation of eTILES algorithm.

Usage

```
etiles(streamfile.edge.removal, init.graph = NULL, obs = 7,
  path = "", start = NULL, end = NULL)
```

Arguments

`streamfile.edge.removal`	.csv file with stream input edges to remove from initial graph
`init.graph`	input initial edge list graph
`obs`	observation window (days)
`path`	Path where generate the results and find the edge file
`start`	starting date
`end`	ending date

Value

dynamic community detection represented in a XXXXXX type of object

See Also

`nchar` which this function wraps

Examples

```
str_length(letters)
```

DynComm$getAlgorithm *getAlgorithm()*

Description

This method returns the algorithm with which this object was initialized.

Usage

```
getAlgorithm()
```

Value

ALGORITHM

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$getAlgorithm()
```

DynComm$nodesAll *nodesAll()*

Description

Get all nodes in the graph after the last iteration.

Usage

```
nodesAll()
```

Value

a list of all nodes in the graph

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$nodesAll()
```

DynComm$nodesCount *nodesCount()*

Description

Get the total number of nodes after the last iteration. It can be useful since nodes can be added, if an edge being added has nodes that do not exist in the graph, or removed, if they are not part of any edge after removing an edge.

Usage

```
nodesCount()
```

Value

an unsigned integer with the number of nodes in the graph

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$nodesCount()
```

DynComm$nodes *nodes(community)*

Description

Get all nodes belonging to the given community after the last iteration.

Usage

```
nodes(community)
```

Arguments

community	The name of the intended community

Value

a list of nodes belonging to the given community

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$nodes(6)
```

DynComm$quality *quality()*

Description

Get the quality measurement of the graph after the last iteration of the algorithm.

Usage

```
quality()
```

Value

a floating point number

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$quality()
```

rdyn *The implementatio of TILES algorithm.*

Description

The implementatio of TILES algorithm.

Usage

```
rdyn(size = 1000, iterations = 100, avg_deg = 15, sigma = 0.6,
  lambdad = 1, alpha = 2.5, paction = 1, prenewal = 0.8,
  quality_threshold = 0.2, new_node = 0, del_node = 0,
  max_evts = 1)
```

Arguments

size	input initial size of graph
iterations	number of iterations
avg_deg	node average degree
sigma	XXXX
lambdad	XXXX
alpha	XXXXX
paction	XXXXX

prenewal	XXXXX
quality_threshold	XXXX
new_node	XXXXX
del_node	XXXXX
max_evts	XXXXX

Value

dynamically generated network in a stream of edges

See Also

nchar which this function wraps

Examples

```
str_length(letters)
```

`tiles` *The implementation of TILES algorithm.*

Description

The implementation of TILES algorithm.

Usage

```
tiles(streamfile, init.graph = NULL, ttl = Inf, obs = 7, path = "",
  start = NULL, end = NULL)
```

Arguments

streamfile	.csv file with stream input edges
init.graph	input initial edge list graph
ttl	edge time to live (days)
obs	observation window (days)
path	Path where generate the results and find the edge file
start	starting date
end	ending date

Value

dynamic community detection represented in a XXXXXX type of object

See Also

nchar which this function wraps

Examples

```
str_length(letters)
```

DynComm$time *time()*

Description

Get the cumulative time spent on processing after the last iteration.

Usage

```
time()
```

Value

an unsigned integer with the total processing time

Examples

```
dc<-DynComm(ALGORITHM$LOUVAIN,QUALITY$MODULARITY,parameters)
dc$time()
## 2.3
```

Index