Package 'robin'

October 30, 2020

Title ROBustness in Network

Version 1.0.2

Maintainer Valeria Policastro <valeria.policastro@gmail.com>

Description
Assesses the robustness of the community structure of a network found by one or more community detection algorithm to give indications about their reliability. It detects if the community structure found by a set of algorithms is statistically significant and compares the different selected detection algorithms on the same network. robin helps to choose among different community detection algorithms the one that better fits the network of interest. Reference in Annamaria Carissimo, Luisa Cutillo, Italia De Feis (2018) <doi:10.1016/j.csda.2017.10.006>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL https://github.com/ValeriaPolicastro/robin

Depends R (>= 3.5), igraph, gprege, qpdf

Imports ggplot2, networkD3, DescTools, fdatest, methods, gridExtra

VignetteBuilder knitr

Suggests devtools, knitr, rmarkdown, testthat (>= 2.1.0)

NeedsCompilation no

Author Valeria Policastro [aut, cre],
Dario Righelli [aut],
Luisa Cutillo [aut],
Italia De Feis [aut],
Annamaria Carissimo [aut]

Repository CRAN

Date/Publication 2020-10-30 14:00:02 UTC

R topics documented:

`membershipCommunities` *membershipCommunities*

Description

This function computes the membership vector of the community structure. To detect the community structure the user can choose one of the methods implemented in igraph.

Usage

```
membershipCommunities(
  graph,
  method = c("walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass",
    "leadingEigen", "labelProp", "infomap", "optimal", "other"),
  FUN = NULL,
  directed = FALSE,
  weights = NULL,
  steps = 4,
  spins = 25,
  e.weights = NULL,
  v.weights = NULL,
  nb.trials = 10
)
```

Arguments

`graph`	The output of prepGraph.
`method`	The clustering method, one of "walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass", "leadingEigen", "labelProp", "infomap", "optimal".

FUN	in case the @method parameter is "other" there is the possibility to use a personal function passing its name through this parameter. The personal parameter has to take as input the @graph and the @weights (that can be NULL), and has to return a community object.
directed	Logical constant, whether to calculate directed edge betweenness for directed graphs. This argument is settable only for "edgeBetweenness" method.
weights	Optional positive weight vector. If the graph has a weight edge attribute, then this is used by default. Supply NA here if the graph has a weight edge attribute, but you want to ignore it. Larger edge weights correspond to stronger connections. This argument is not settable for "infomap" method.
steps	The number of steps to take, this is actually the number of tries to make a step. It is not a particularly useful parameter. This argument is settable only for "leadingEigen"and"walktrap" method.
spins	Integer constant, the number of spins to use. This is the upper limit for the number of communities. It is not a problem to supply a (reasonably) big number here, in which case some spin states will be unpopulated. This argument is settable only for "spinglass" method.
e.weights	If not NULL, then a numeric vector of edge weights. The length must match the number of edges in the graph. By default the 'weight' edge attribute is used as weights. If it is not present, then all edges are considered to have the same weight. Larger edge weights correspond to stronger connections. This argument is settable only for "infomap" method.
v.weights	If not NULL, then a numeric vector of vertex weights. The length must match the number of vertices in the graph. By default the 'weight' vertex attribute is used as weights. If it is not present, then all vertices are considered to have the same weight. A larger vertex weight means a larger probability that the random surfer jumps to that vertex. This argument is settable only for "infomap" method.
nb.trials	The number of attempts to partition the network (can be any integer value equal or larger than 1). This argument is settable only for "infomap" method.

Value

Returns a numeric vector, one number for each vertex in the graph; the membership vector of the community structure.

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
membershipCommunities (graph=graph, method="louvain")
```

methodCommunity *methodCommunity*

Description

This function detects the community structure of a graph. To detect the community structure the user can choose one of the methods implemented in igraph.

Usage

```
methodCommunity(
  graph,
  method = c("walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass",
    "leadingEigen", "labelProp", "infomap", "optimal", "other"),
  FUN = NULL,
  directed = FALSE,
  weights = NULL,
  steps = 4,
  spins = 25,
  e.weights = NULL,
  v.weights = NULL,
  nb.trials = 10,
  verbose = FALSE
)
```

Arguments

graph	The output of prepGraph.
method	The clustering method, one of "walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass", "leadingEigen", "labelProp", "infomap", "optimal", "other".
FUN	in case the @method parameter is "other" there is the possibility to use a personal function passing its name through this parameter. The personal parameter has to take as input the @graph and the @weights (that can be NULL), and has to return a community object.
directed	Logical constant, whether to calculate directed edge betweenness for directed graphs. This argument is settable only for "edgeBetweenness" method.
weights	Optional positive weight vector. If the graph has a weight edge attribute, then this is used by default. Supply NA here if the graph has a weight edge attribute, but you want to ignore it. Larger edge weights correspond to stronger connections. This argument is not settable for "infomap" method.
steps	The number of steps to take, this is actually the number of tries to make a step. It is not a particularly useful parameter. This argument is settable only for "leadingEigen" and "walktrap" method.

spins	Integer constant, the number of spins to use. This is the upper limit for the number of communities. It is not a problem to supply a (reasonably) big number here, in which case some spin states will be unpopulated. This argument is settable only for "spinglass" method.
e.weights	If not NULL, then a numeric vector of edge weights. The length must match the number of edges in the graph. By default the 'weight' edge attribute is used as weights. If it is not present, then all edges are considered to have the same weight. Larger edge weights correspond to stronger connections. This argument is settable only for "infomap" method.
v.weights	If not NULL, then a numeric vector of vertex weights. The length must match the number of vertices in the graph. By default the 'weight' vertex attribute is used as weights. If it is not present, then all vertices are considered to have the same weight. A larger vertex weight means a larger probability that the random surfer jumps to that vertex. This argument is settable only for "infomap" method.
nb.trials	The number of attempts to partition the network (can be any integer value equal or larger than 1). This argument is settable only for "infomap" method.
verbose	flag for verbose output (default as FALSE)

Value

A Communities object.

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
methodCommunity (graph=graph, method="louvain")
```

---

plotComm *plotComm*

Description

Graphical interactive representation of the network and its communities.

Usage

```
plotComm(graph, members)
```

Arguments

graph	The output of prepGraph.
members	A membership vector of the community structure, the output of membership-Communities.

Value

Creates an interactive plot with colorful communities, a D3 JavaScript network graph.

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
members <- membershipCommunities (graph=graph, method="louvain")
plotComm(graph, members)
```

---

plotGraph *plotGraph*

Description

Graphical interactive representation of the network.

Usage

```
plotGraph(graph)
```

Arguments

graph	The output of prepGraph.

Value

Creates an interactive plot, a D3 JavaScript network graph.

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
plotGraph (graph)
```

plotRobin *plotRobin*

Description

This function plots two curves: the measure of the null model and the measure of the real graph or the measure of the two community detection algorithms.

Usage

```
plotRobin(
  graph,
  model1,
  model2,
  measure = c("vi", "nmi", "split.join", "adjusted.rand"),
  legend = c("model1", "model2"),
  title = "Robin plot"
)
```

Arguments

graph	The output of prepGraph
model1	The Mean output of the robinRobust function or the Mean1 output of robinCompare.
model2	The MeanRandom output of the robinRobust function or the Mean2 output of robinCompare.
measure	The stability measure: one of "vi", "nmi", "split.join", "adjusted.rand".
legend	The legend for the graph. The default is c("model1", "model2"). If using robinRobust is recommended c("real data", "null model"), if using robinCompare, enter the names of the community detection algorithms.
title	The title for the graph. The default is "Robin plot".

Value

A ggplot object.

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
graphRandom <- random(graph=graph)
Proc <- robinRobust(graph=graph, graphRandom=graphRandom, method="louvain",
type="independent")
plotRobin(graph=graph, model1=Proc$Mean, model2=Proc$MeanRandom,
measure="vi", legend=c("real data", "null model"))
```

prepGraph *prepGraph*

Description

This function reads graphs from a file and prepares them for the analysis.

Usage

```
prepGraph(
  file,
 file.format = c("edgelist", "pajek", "ncol", "lgl", "graphml", "dimacs", "graphdb",
    "gml", "dl", "igraph"),
  numbers = FALSE,
  directed = FALSE,
  header = FALSE,
  verbose = FALSE
)
```

Arguments

file	The input file containing the graph.
file.format	Character constant giving the file format. Edgelist, pajek, graphml, gml, ncol, lgl, dimacs, graphdb and igraph are supported.
numbers	A logical value indicating if the names of the nodes are values.This argument is settable for the edgelist format. The default is FALSE.
directed	A logical value indicating if is a directed graph. The default is FALSE.
header	A logical value indicating whether the file contains the names of the variables as its first line.This argument is settable
verbose	flag for verbose output (default as FALSE). for the edgelist format.The default is FALSE.

Value

An igraph object, which do not contain loop and multiple edges.

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
```

random *random*

Description

This function randomly rewires the edges while preserving the original graph's degree distribution.

Usage

```
random(graph, verbose = FALSE)
```

Arguments

graph	The output of prepGraph.
verbose	flag for verbose output (default as FALSE)

Value

An igraph object, a randomly rewired graph.

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
graphRandom <- random(graph=graph)
```

robinAUC *robinAUC*

Description

This function calculates the area under two curves with a spline approach.

Usage

```
robinAUC(
  graph,
  model1,
  model2,
  measure = c("vi", "nmi", "split.join", "adjusted.rand"),
  verbose = FALSE
)
```

Arguments

graph	The output of prepGraph.
model1	The Mean output of the robinRobust function (or the Mean1 output of the robinCompare function).
model2	The MeanRandom output of the robinRobust function (or the Mean2 output of the robinCompare function).
measure	The stability measure "vi", "nmi", "split.join", "adjusted.rand".
verbose	flag for verbose output (default as FALSE).

Value

A list

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
graphRandom <- random(graph=graph)
Proc <- robinRobust(graph=graph, graphRandom=graphRandom, method="louvain",
measure="vi",type="independent")
robinAUC(graph=graph, model1=Proc$Mean, model2=Proc$MeanRandom)
```

---

robinCompare *robinCompare*

Description

This function compares the robustness of two community detection algorithms.

Usage

```
robinCompare(
  graph,
  method1 = c("walktrap", "edgeBetweenness", "fastGreedy", "leadingEigen", "louvain",
    "spinglass", "labelProp", "infomap", "optimal", "other"),
  method2 = c("walktrap", "edgeBetweenness", "fastGreedy", "leadingEigen", "louvain",
    "spinglass", "labelProp", "infomap", "optimal", "other"),
  FUN1 = NULL,
  FUN2 = NULL,
  measure = c("vi", "nmi", "split.join", "adjusted.rand"),
  type = c("independent", "dependent"),
  directed = FALSE,
  weights = NULL,
  steps = 4,
  spins = 25,
  e.weights = NULL,
```

```
  v.weights = NULL,
  nb.trials = 10,
  verbose = FALSE
)
```

Arguments

graph	The output of prepGraph.
method1	The first clustering method, one of "walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass", "leadingEigen", "labelProp", "infomap","optimal".
method2	The second custering method one of "walktrap", "edgeBetweenness","fastGreedy", "louvain", "spinglass", "leadingEigen", "labelProp", "infomap","optimal".
FUN1	personal designed function when method1 is "others". see methodCommunity.
FUN2	personal designed function when method2 is "others". see methodCommunity.
measure	The stability measure, one of "vi", "nmi", "split.join", "adjusted.rand".
type	The type of robin costruction, dependent or independent.
directed	This argument is settable only for "edgeBetweenness" method.
weights	This argument is not settable for "infomap" method.
steps	This argument is settable only for "leadingEigen"and"walktrap" method.
spins	This argument is settable only for "infomap" method.
e.weights	This argument is settable only for "infomap" method.
v.weights	This argument is settable only for "infomap" method.
nb.trials	This argument is settable only for "infomap" method.
verbose	flag for verbose output (default as FALSE).

Value

A list object with two matrices: - the matrix "Mean1" with the means of the procedure for the first method - the matrix "Mean2" with the means of the procedure for the second method.

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
robinCompare(graph=graph, method1="louvain",
method2="fastGreedy", measure="vi", type="independent")
```

robinFDATest *robinFDATest*

Description

The function implements the Interval Testing Procedure to test the difference between two curves.

Usage

```
robinFDATest(
  graph,
  model1,
  model2,
  measure = c("vi", "nmi", "split.join", "adjusted.rand"),
  legend = c("model1", "model2"),
  verbose = FALSE
)
```

Arguments

graph	The output of prepGraph.
model1	The Mean output of the robinRobust function (or the Mean1 output of the robinCompare function).
model2	The MeanRandom output of the robinRobust function (or the Mean2 output of the robinCompare function).
measure	The stability measure "vi", "nmi", "split.join", "adjusted.rand".
legend	The legend for the graph. The default is c("model1", "model2"). If using robinRobust is recommended c("real data", "null model"), if using robinCompare, enter the names of the community detection algorithms.
verbose	flag for verbose output (default as FALSE).

Value

Two plots: the fitted curves and the adjusted p-values. A vector of the adjusted p-values.

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
graphRandom <- random(graph=graph)
Proc <- robinRobust(graph=graph, graphRandom=graphRandom, method="louvain",
measure="vi",type="independent")
robinFDATest(graph=graph, model1=Proc$Mean, model2=Proc$MeanRandom,
measure="vi",legend=c("real data", "null model"))
```

robinGPTest *robinGPTest*

Description

This function implements the GP testing procedure and calculates the Bayes factor.

Usage

```
robinGPTest(model1, model2, verbose = FALSE)
```

Arguments

model1	The Mean output of the robinRobust function (or the Mean1 output of the robinCompare function).
model2	The MeanRandom output of the robinRobust function (or the Mean2 output of the robinCompare function).
verbose	flag for verbose output (default as FALSE).

Value

A numeric value, the Bayes factor

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
graphRandom <- random(graph=graph)
Proc <- robinRobust(graph=graph, graphRandom=graphRandom,
method="louvain", measure="vi",type="independent")
robinGPTest(model1=Proc$Mean,model2=Proc$MeanRandom)
```

robinRobust *robinRobust*

Description

This functions implements a procedure to examine the stability of the partition recovered by some algorithm against random perturbations of the original graph structure.

Usage

```
robinRobust(
  graph,
  graphRandom,
  method = c("walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass",
    "leadingEigen", "labelProp", "infomap", "optimal", "other"),
  FUN = NULL,
  measure = c("vi", "nmi", "split.join", "adjusted.rand"),
  type = c("independent", "dependent"),
  directed = FALSE,
  weights = NULL,
  steps = 4,
  spins = 25,
  e.weights = NULL,
  v.weights = NULL,
  nb.trials = 10,
  verbose = FALSE
)
```

Arguments

graph	The output of prepGraph.
graphRandom	The output of random function.
method	The clustering method, one of "walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass", "leadingEigen", "labelProp", "infomap", "optimal".
FUN	in case the @method parameter is "other" there is the possibility to use a personal function passing its name through this parameter. The personal parameter has to take as input the @graph and the @weights (that can be NULL), and has to return a community object.
measure	The stability measure, one of "vi", "nmi", "split.join", "adjusted.rand".
type	The type of robin costruction, dependent or independent data
directed	This argument is settable only for "edgeBetweenness" method.
weights	this argument is not settable for "infomap" method.
steps	this argument is settable only for "leadingEigen"and"walktrap" method.
spins	This argument is settable only for "infomap" method.
e.weights	This argument is settable only for "infomap" method.
v.weights	This argument is settable only for "infomap" method.
nb.trials	This argument is settable only for "infomap" method.
verbose	flag for verbose output (default as FALSE).

Value

A list object with two matrices: - the matrix "Mean" with the means of the procedure for the graph - the matrix "MeanRandom" with the means of the procedure for the random graph.

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
graphRandom <- random(graph=graph)
robinRobust(graph=graph, graphRandom=graphRandom, method="louvain",
measure="vi",type="independent")
```

Index