Package 'robin'

October 24, 2019

Title ROBustness in Network

Version 0.99.1

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

Description Many community detection algorithms have been developed in network analysis. However, their applications leave unaddressed the statistical validation of the results, for this reason we developed ROBIN (ROBustness In Network), a useful method for the validation of community detection. It has a double aim, it studies the robustness of a single community detection algorithm and compares two community detection algorithms to understand which provides the best partition. 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 6.1.1

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

Depends R (>= 3.5), igraph, gprege

Imports ggplot2, networkD3, DescTools, fdatest, methods

VignetteBuilder knitr

Suggests devtools, cowplot, 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 2019-10-24 18:10:05 UTC

R topics documented:

`membershipCommunities` *membershipCommunities*

Description

This function gives the membership vector of the community structure. The community structure are found by all functions 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`	see `methodCommunity`.
`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. The community structure are found by all functions 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)
```

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), moreover it has to return a communities 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

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 simple D3 JavaScript network graph,an interactive plot.

Examples

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

---

plotRobin *plotRobin*

Description

The plot of the two curves, the measures of the null model and of the real graph.

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 "vi", "nmi", "split.join", "adjusted.rand".
legend	The legend for the graph. The default is c("model1", "model2").
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

The prepGraph function is able to read graphs from a file and to prepare 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)
```

Arguments

file	The file to read from.
file.format	Character constant giving the file format. Right now as_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 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

Randomly rewire the edges while preserving the original graph's degree distribution.

Usage

```
random(graph)
```

Arguments

graph	The output of prepGraph.

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

Calculate the area under the two curves with a spline approach.

Usage

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

Arguments

graph	The output of prepGraph.
model1	The Mean output of the robinRobust function (or the Mean1 output of the comparison function).
model2	The MeanRandom output of the robinRobust function (or the Mean2 output of the comparison function).
measure	The stability measure "vi", "nmi", "split.join", "adjusted.rand".

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

A procedure to compare two different methods of community detection.

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)
```

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	its a personal designed function when method1 is "others". see methodCommunity.
FUN2	its a personal designed function when method2 is "others". see methodCommunity.
measure	The stability measure "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.

Value

A list object

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 for testing the difference between the two curves.

Usage

```
robinFDATest(graph, model1, model2, measure = c("vi", "nmi",
  "split.join", "adjusted.rand"), legend = c("real data", "null model"))
```

Arguments

graph	The output of prepGraph.
model1	The Mean output of the robinRobust function (or the Mean1 output of the comparison function).
model2	The MeanRandom output of the robinRobust function (or the Mean2 output of the comparison function).
measure	The stability measure "vi", "nmi", "split.join", "adjusted.rand".
legend	The legend for the graph. The default is c("real data", "null model").

Value

Two plots.

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)
```

robinGPTest *robinGPTest*

Description

This function makes a test between the curves, calculating the Bayes factor.

Usage

```
robinGPTest(ratio)
```

Arguments

ratio	The ratios output of the robinRobust function (or the ratios1vs2 output of the comparison function).

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(ratio=Proc$ratios)
```

robinRobust *robinRobust*

Description

A procedure to examine the stability of the partition recovered 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)
```

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	see methodCommunity.
measure	The stability measure "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.

Value

A list object.

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