Package ‘robin’

July 23, 2025

Title ROBustness in Network

Version 2.0.0

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 Policastro V., Righelli D., Carissimo A., Cutillo L., De Feis I. (2021) <https://journal.r-project.org/archive/2021/RJ-2021-040/index.html>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

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

Depends igraph

Imports ggplot2, networkD3, DescTools, fdatest, methods, gridExtra, spam, qpdf, Matrix, perturbR, BiocParallel, reshape2

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 2025-01-22 17:30:10 UTC

Contents

`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", "leiden", "other"),
  ...,
  FUN = NULL
)
```

Arguments

`graph`	The output of prepGraph.
`method`	The clustering method, one of "walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass", "leadingEigen", "labelProp", "infomap", "optimal", "leiden","other".
`...`	additional parameters to use with any of the previous described methods (see igraph package community detection methods for more details i.e. cluster_walktrap)
`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.

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", "leiden", "other"),
  leiden_objective_function = c("modularity", "CPM"),
  ...,
  FUN = NULL,
  verbose = FALSE
)
```

Arguments

`graph`	The output of prepGraph.
`method`	The clustering method, one of "walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass", "leadingEigen", "labelProp", "infomap", "optimal", "leiden","other".
`leiden_objective_function`	objective_function parameter for leiden only for method
`...`	additional parameters to use with any of the previous described methods (see igraph package community detection methods for more details i.e. cluster_walktrap)
`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.
`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")
```

`plot.robin` *plot.robin*

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

```
## S3 method for class 'robin'
plot(x, title = "Robin plot", ...)
```

Arguments

`x`	A robin class object. The output of the functions: `robinRobust` and `robinCompare`.
`title`	The title for the graph. The default is "Robin plot".
`...`	other parameter

Value

A ggplot object.

Examples

```
## Not run: my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
comp <- robinCompare(graph=graph, method1="fastGreedy",method2="louvain")
plot(comp)
## End(Not run)
```

`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 membershipCommunities.

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)
```

`plotMultiCompare` *plotMultiCompare*

Description

This function plots the curves of the measure of many community detection algorithms compared.

Usage

```
plotMultiCompare(..., title = "Robin plot", ylim1 = FALSE)
```

Arguments

`...`	all robin objects obtained from the comparison between one community detection algorithm and all the others
`title`	character a title for the plot (default is "Robin plot")
`ylim1`	logical for spanning the y axis from 0 to 1 (default is FALSE)

Value

a ggplot2 object

Examples

```
my_file <- system.file("example/football.gml", package="robin")
graph <- prepGraph(file=my_file, file.format="gml")
comp1 <- robinCompare(graph=graph, method1="fastGreedy",method2="louvain")
comp2 <- robinCompare(graph=graph, method1="fastGreedy",method2="infomap")
plotMultiCompare(comp1,comp2)
```

`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

```
#install.packages("robin")

#If there are problems with the installation try:
# if (!requireNamespace("BiocManager", quietly = TRUE))
#     install.packages("BiocManager")
# BiocManager::install("gprege")
# install.packages("robin")

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, dist = "NegBinom", verbose = FALSE)
```

Arguments

`graph`	The output of prepGraph.
`dist`	Option to rewire in a manner that retains overall graph weight regardless of distribution of edge weights. This option is invoked by putting any text into this field. Defaults to "NegBinom" for negative binomial.
`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(x, verbose = FALSE)
```

Arguments

`x`	A robin class object. The output of the functions: `robinRobust` and `robinCompare`.
`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")
robinAUC(proc)
```

`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", "leiden", "other"),
  args1 = list(),
 method2 = c("walktrap", "edgeBetweenness", "fastGreedy", "leadingEigen", "louvain",
    "spinglass", "labelProp", "infomap", "optimal", "leiden", "other"),
  args2 = list(),
  FUN1 = NULL,
  FUN2 = NULL,
  measure = c("vi", "nmi", "split.join", "adjusted.rand"),
  type = "independent",
  verbose = TRUE,
  dist = "Other",
  BPPARAM = BiocParallel::bpparam()
)
```

Arguments

`graph`	The output of prepGraph.
`method1`	The first clustering method, one of "walktrap", "edgeBetweenness", "fastGreedy", "louvain", "spinglass", "leadingEigen", "labelProp", "infomap","leiden","optimal","other".
`args1`	A `list` of arguments to be passed to the `method1` (see i.e. cluster_leiden for a list of possible method parameters).
`method2`	The second custering method one of "walktrap", "edgeBetweenness","fastGreedy", "louvain", "spinglass", "leadingEigen", "labelProp", "infomap","leiden","optimal","other".

`args2`	A `list` of arguments to be passed to the `method2` (see i.e. cluster_leiden for a list of possible method parameters).
`FUN1`	personal designed function when `method1` is "other". see `methodCommunity`.
`FUN2`	personal designed function when `method2` is "other". see `methodCommunity`.
`measure`	The stability measure, one of "vi", "nmi", "split.join", "adjusted.rand" all normalized and used as distances. "nmi" refers to 1- nmi and "adjusted.ran" refers to 1-adjusted.rand.
`type`	The type of robin construction, dependent or independent.
`verbose`	flag for verbose output (default as TRUE).
`dist`	Option to rewire in a manner that retains overall graph weight regardless of distribution of edge weights. This option is invoked by putting any text into this field. Defaults to "Other". See `rewireR` for details.
`BPPARAM`	the BiocParallel object of class `bpparamClass` that specifies the back-end to be used for computations. See `bpparam` for details.

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", args1 = list(resolution=0.8),
            method2="leiden")
```

`robinFDATest` *robinFDATest*

Description

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

Usage

```
robinFDATest(x, verbose = FALSE)
```

Arguments

`x`	A robin class object. The output of the functions: `robinRobust` and `robinCompare`.
`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")
comp <- robinCompare(graph=graph, method1="fastGreedy",method2="infomap")
robinFDATest(comp)
```

`robinGPTest` *robinGPTest*

Description

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

Usage

```
robinGPTest(x, verbose = FALSE)
```

Arguments

`x`	A robin class object. The output of the functions: `robinRobust` and `robinCompare`.
`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")
comp <- robinCompare(graph=graph, method1="fastGreedy",method2="infomap")
robinGPTest(comp)
```

`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", "leiden", "other"),
  ...,
  FUN = NULL,
  measure = c("vi", "nmi", "split.join", "adjusted.rand"),
  type = "independent",
  verbose = TRUE,
  dist = "Other",
  BPPARAM = BiocParallel::bpparam()
)
```

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", "leiden","optimal".
`...`	other parameter.
`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" all normalized and used as distances. "nmi" refers to 1- nmi and "adjusted.ran" refers to 1-adjusted.rand.
`type`	The type of robin construction, dependent or independent.
`verbose`	flag for verbose output (default as TRUE).
`dist`	Option to rewire in a manner that retains overall graph weight regardless of distribution of edge weights. This option is invoked by putting any text into this field. Defaults to "Other". See `rewireR` for details.
`BPPARAM`	the BiocParallel object of class `bpparamClass` that specifies the back-end to be used for computations. See `bpparam` for details.

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="leiden")
```

Index