Package 'statGraph'

December 16, 2020

Type Package

Title Statistical Methods for Graphs

Author Diogo R. da Costa [aut],
Taiane C. Ramos [aut],
Grover E. Castro Guzman [aut],
Suzana S. Santos [aut],
Eduardo S. Lira [aut],
Andre Fujita [aut, cre]

Maintainer Andre Fujita <andrefujita@usp.br>

Depends R (>= 3.6.0), stats, graphics

Imports igraph, MASS, rARPACK, cluster, foreach, parallel, doParallel

Description Contains statistical methods to analyze graphs, such as graph parameter estimation, model selection based on the GIC (Graph Information Criterion), statistical tests to discriminate two or more populations of graphs (ANOGVA - Analysis of Graph Variability), correlation between graphs, and clustering of graphs.
References: Takahashi et al. (2012) <doi:10.1371/journal.pone.0049949>,
Futija et al. (2017) <doi:10.3389/fnins.2017.00066>,
Fujita et al. (2017) <doi:10.1016/j.csda.2016.11.016>,
Tang et al. (2017) <doi:10.3150/15-BEJ789>,
Tang et al. (2017) <doi:10.1080/10618600.2016.1193505>,
Ghoshdastidar et al. (2017) <arXiv:1705.06168>,
Ghoshdastidar et al. (2017) <arXiv:1707.00833>,
Cerqueira et al. (2017) <doi:10.1109/TNSE.2017.2674026>,
Fraiman and Fraiman (2018) <doi:10.1038/s41598-018-23152-5>,
Fujita et al. (2019) <doi:10.1093/comnet/cnz028>.

License GPL (>= 3)

Encoding UTF-8

LazyLoad yes

URL https://www.ime.usp.br/~fujita/software.html

Version 0.4.1

Date 2020-12-14

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2020-12-16 22:30:02 UTC

R topics documented:

`statGraph-package` *Statistical Methods for Graphs*

Description

Contains statistical methods to analyze graphs, such as graph parameter estimation, model selection based on the GIC (Graph Information Criterion), statistical tests to discriminate two or more populations of graphs (ANOGVA - Analysis of Graph Variability), correlation between graphs, and clustering of graphs. References: Takahashi et al. (2012) <doi:10.1371/journal.pone.0049949>, Futija et al. (2017) <doi:10.3389/fnins.2017.00066>, Fujita et al. (2017) <doi:10.1016/j.csda.2016.11.016>, Tang et al. (2017) <doi:10.3150/15-BEJ789>, Tang et al. (2017) <doi:10.1080/10618600.2016.1193505>, Ghoshdastidar et al. (2017) <arXiv:1705.06168>, Ghoshdastidar et al. (2017) <arXiv:1707.00833>, Cerqueira et al. (2017) <doi:10.1109/TNSE.2017.2674026>, Fraiman and Fraiman (2018) <doi:10.1038/s41598-018-23152-5>, Fujita et al. (2019) <doi:10.1093/comnet/cnz028>.

Details

The DESCRIPTION file:

Package:	statGraph
Type:	Package
Version:	0.4.1
Date:	2020-12-16
Depends:	R (>= 3.6.0), stats, graphics
Imports:	igraph, MASS, rARPACK, cluster, foreach, parallel, doParallel
Encoding:	UTF-8
License:	GPL (>= 3)
LazyLoad:	yes
URL:	https://www.ime.usp.br/~fujita/software.html

Author(s)

Diogo R. da Costa [aut], Taiane C. Ramos [aut], Grover E. Castro Guzman [aut], Suzana S. Santos [aut], Eduardo S. Lira [aut], Andre Fujita [aut, cre]

Maintainer: Andre Fujita <andrefujita@usp.br>

See Also

Useful links:

- https://www.ime.usp.br/~fujita/software.html

`anogva` *ANOGVA Analysis Of Graph Variability*

Description

`anogva` statistically tests whether two or more sets of graphs are generated by the same random graph model. It is a generalization of the 'graph.test' function.

Usage

```
anogva(graphs, labels, numBoot = 1000, bandwidth = "Silverman")
```

Arguments

`graphs`	a list of adjacency (symmetric) matrices of undirected graphs. For unweighted graphs, each matrix contains only 0s and 1s. For weighted graphs, each matrix may contain real values that correspond to the weights of the edges.
`labels`	an array of integers indicating the labels of each graph.
`numBoot`	integer indicating the number of bootstrap resamplings.

bandwidth	string indicating which criterion will be used to choose the bandwidth for the spectral density estimation. The available criteria are "Silverman" (default) and "Sturges".

Value

A list containing:

statistic	the statistic of the test.
p.value	the p-value of the test.

References

Fujita, A., Vidal, M. C. and Takahashi, D. Y. (2017) A Statistical Method to Distinguish Functional Brain Networks. _Front. Neurosci._, *11*, 66. doi:10.3389/fnins.2017.00066.

Takahashi, D. Y., Sato, J. R., Ferreira, C. E. and Fujita A. (2012) Discriminating Different Classes of Biological Networks by Analyzing the Graph Spectra Distribution. _PLoS ONE_, *7*, e49949. doi:10.1371/journal.pone.0049949.

Silverman, B. W. (1986) _Density Estimation_. London: Chapman and Hall.

Sturges, H. A. The Choice of a Class Interval. _J. Am. Statist. Assoc._, *21*, 65-66.

Examples

```
require(igraph)
g1 <- g2 <- g3 <- list()
for (i in 1:20) {
   G1 <- erdos.renyi.game(50, 0.50)
   g1[[i]] <- get.adjacency(G1)
   G2 <- erdos.renyi.game(50, 0.50)
   g2[[i]] <- get.adjacency(G2)
   G3 <- erdos.renyi.game(50, 0.52)
   g3[[i]] <- get.adjacency(G3)
}
g <- c(g1, g2, g3)
label <- c(rep(1,20),rep(2,20),rep(3,20))
result <- anogva(g, label, numBoot=50)
result
```

`cerqueira` *Andressa Cerqueira, Daniel Fraiman, Claudia D. Vargas and Florencia Leonardi non-parametric test of hypotheses to verify if two samples of random graphs were originated from the same probability distribution.*

Description

Given two identically independently distributed (idd) samples of graphs g and gp, the test verifies if they have the same distribution by calculating the mean distance D from g to gp. The test rejects the null hypothesis if D is greater than the (1-alpha)-quantile of the distribution of the test under the null hypothesis.

Usage

```
cerqueira(g, gp, maxPer = 300, alpha = 0.05, printResult = FALSE)
```

Arguments

g	the first iid sample of graphs to be compared. Must be a list of igraph objects.
gp	the second iid sample of graphs to be compared. Must be a list of igraph objects.
maxPer	integer indicating the number of bootstrap resamples (default is 300).
alpha	the significance level for the test (default is 0.05).
printResult	logical indicating if the test must print the result (default is FALSE).

Value

A list containing:

test_stats	the value of the test.
p_value	the p-value of the test.
reject_threshold	The 1-alpha quantile of the test distribution under the null hypothesys.
bootstrap_samples	The test distrition on the bootstrap resamples.

References

Andressa Cerqueira, Daniel Fraiman, Claudia D. Vargas and Florencia Leonardi. "A test of hypotheses for random graph distributions built from EEG data", https://ieeexplore.ieee.org/document/7862892

Examples

```
## Not run:
require(igraph)
set.seed(42)

## test under H0
a <- b <- list()
for(i in 1:10){
  a[[i]] <- erdos.renyi.game(50,0.5)
  b[[i]] <- erdos.renyi.game(50,0.5)
}
k <- cerqueira(a, b, printResult = TRUE)
```

```
## test under H1
a <- b <- list()
for(i in 1:10){
  a[[i]] <- erdos.renyi.game(50,0.5)
  b[[i]] <- erdos.renyi.game(50,0.6)
}
k <- cerqueira(a, b, printResult = TRUE)

## End(Not run)
```

`fast.eigenvalue.probability` *Degree-based eigenvalue probability*

Description

`fast.eigenvalue.probability` returns the probability of an eigenvalue given the degree and excess degree probability.

Usage

```
fast.eigenvalue.probability(deg_prob, q_prob, all_k, z, n_iter = 5000)
```

Arguments

`deg_prob`	The degree probability of the graph.
`q_prob`	The excess degree probability of the graph.
`all_k`	List of sorted unique degrees greater than 1 of the graph.
`z`	Complex number whose real part is the eigenvalue whose probability we want to obtain, the imaginary part is a small value (e.g., 1e-3).
`n_iter`	The maximum number of iterations to perform.

Value

A complex number whose imaginary part absolute value corresponds to the probability of the given eigenvalue.

References

Newman, M. E. J., Zhang, X., & Nadakuditi, R. R. (2019). Spectra of random networks with arbitrary degrees. Physical Review E, 99(4), 042309.

Examples

```
G <- igraph::sample_smallworld(dim = 1, size = 10, nei = 2, p = 0.2)
# Obtain the degree distribution
deg_prob <- c(igraph::degree_distribution(graph = G, mode = "all"),0.0)
k_deg <- seq(1,length(deg_prob)) - 1
# Obtain the excess degree distribution
c <- sum(k_deg * deg_prob)
q_prob <- c()
for(k in 0:(length(deg_prob) - 1)){
  aux_q <- (k + 1) * deg_prob[k + 1]/c
  q_prob <- c(q_prob,aux_q)
}
# Obtain the sorted unique degrees greater than 1
all_k <- c(1:length(q_prob))
valid_idx <- q_prob != 0
q_prob <- q_prob[valid_idx]
all_k <- all_k[valid_idx]
# Obtain the probability of the eigenvalue 0
z <- 0 + 0.01*1i
eigenval_prob <- -Im(fast.eigenvalue.probability(deg_prob,q_prob,all_k,z))
eigenval_prob
```

`fast.graph.param.estimator`

Degree-based graph parameter estimator

Description

`fast.graph.param.estimator` estimates the parameter of the complex network model using the degree-based spectral density and ternary search.

Usage

```
fast.graph.param.estimator(
  graph,
  model,
  lo = NULL,
  hi = NULL,
  eps = 0.001,
  from = NULL,
  to = NULL,
  npoints = 2000,
  numCores = 1
)
```

Arguments

graph	The undirected unweighted graph (igraph type).
model	Either a string or a function: A string that indicates one of the following models: "ER" (Erdos-Renyi random graph model), "GRG" (geometric random graph model), "WS" (Watts-Strogatz model), and "BA" (Barabasi-Albert model). A function that returns a Graph generated by a graph model. It must contain two arguments: the first one corresponds to the graph size and the second to the parameter of the model.
lo	Smallest parameter value that the graph model can take. If "model" is an string, then the default value of 0 is used for the predefined models ("ER", "GRG", "WS", and "BA").
hi	Largest parameter value that the graph model can take. If "model" is an string, then the default values are used for the predefined models 1 for "ER", sqrt(2) for "GRG", 1 for "WS", and 5 for "BA").
eps	Desired precision of the parameter estimate.
from	Lower end of the interval that contain the eigenvalues to generate the degree-based spectral densities. The smallest eigenvalue of the adjacency matrix corresponding to "graph" is used if the value is not given.
to	Upper end of the interval that contain the eigenvalues to generate the degree-based spectral densities. The largest eigenvalue of the adjacency matrix corresponding to "graph" is used if the value is not given.
npoints	Number of points to discretize the interval <from,to>.
numCores	Number of cores to use for parallelization.

Value

Returns a list containing:

param	The degree-based parameter estimate. For the "ER", "GRG", "WS", and "BA" models, the parameter corresponds to the probability to connect a pair of vertices, the radius used to construct the geometric graph in a unit square, the probability to reconnect a vertex, and the scaling exponent respectively.
dist	The L1 distance between the observed graph and the graph model with the estimated value.

Examples

```
### Example giving only the name of the model to use
G <- igraph::sample_smallworld(dim = 1, size = 15, nei = 2, p = 0.2)
# Obtain the parameter of the WS model
estimated.parameter <- fast.graph.param.estimator(G, "WS",lo = 0,hi = 0.5,eps = 1e-1, npoints = 10,
                                                  numCores = 1)
estimated.parameter
### Example giving a function instead of a model (uncomment to execute)
# Defining the model to use
```

```
#G <- igraph::sample_smallworld(dim = 1, size = 5000, nei = 2, p = 0.2)
#K <- as.integer(igraph::ecount(G)/igraph::vcount(G))
#fun_WS <- function(n, param, nei = K){
# return (igraph::sample_smallworld(dim = 1,size = n, nei = nei,p = param))
#}
# Obtain the parameter of the WS model
#estimated.parameter <- fast.graph.param.estimator(G, fun_WS, lo = 0.0, hi = 1.0,
#                                                  npoints = 100, numCores = 2)
#estimated.parameter
```

`fast.spectral.density` *Degree-based spectral density*

Description

`fast.spectral.density` returns the degree-based spectral density in the interval <`from`,`to`> by using npoints discretization points.

Usage

```
fast.spectral.density(
  graph,
  from = NULL,
  to = NULL,
  npoints = 2000,
  numCores = 1
)
```

Arguments

`graph`	The undirected unweighted graph (igraph type) whose spectral density we want to obtain.
`from`	Lower end of the interval that contain the eigenvalues or smallest eigenvalue of the adjacency matrix of the graph. The smallest eigenvalue is used if the value is not given.
`to`	Upper end of the interval that contain the eigenvalues or largest eigenvalue of the adjacency matrix of the graph. The largest eigenvalue is used if the value is not given.
`npoints`	Number of discretization points of the interval <`from`,`to`>.
`numCores`	Number of cores to use for parallelization.

Value

Returns the degree-based spectral density of the graph in the

References

Newman, M. E. J., Zhang, X., & Nadakuditi, R. R. (2019). Spectra of random networks with arbitrary degrees. Physical Review E, 99(4), 042309.

Examples

```
G <- igraph::sample_smallworld(dim = 1, size = 100, nei = 2, p = 0.2)
# Obtain the degree-based spectral density
density <- fast.spectral.density(graph = G, npoints = 80, numCores = 1)
density
```

`fraiman` *Daniel Fraiman and Ricardo Fraiman test for network differences between groups with an analysis of variance test (ANOVA).*

Description

Given a list of graphs, the test verifies if all the subpopulations have the same mean network, under the alternative that at least one subpopulation has a different mean network.

Usage

```
fraiman(g, maxPer = 300, alpha = 0.05, printResult = FALSE)
```

Arguments

`g`	the undirected graphs to be compared. Must be a list of lists of igraph objects or a list of lists of adjacency matrices.
`maxPer`	integer indicating the number of bootstrap resamples (default is 300).
`alpha`	the significance level for the test (default is 0.05).
`printResult`	logical indicating if the test must print the result (default is FALSE).

Value

A list containing:

`test_stats`	the value of the test.
`p_value`	the p-value of the test.
`bootstrap_samples`	The test distrition on the bootstrap resamples.

References

Fraiman, Daniel, and Ricardo Fraiman. "An ANOVA approach for statistical comparisons of brain networks", https://www.nature.com/articles/s41598-018-23152-5

Examples

```
## Not run:
require(igraph)
set.seed(42)

## test under H0
a <- b <- d <- list()
for(i in 1:10){
  a[[i]] <- erdos.renyi.game(50,0.5)
  b[[i]] <- erdos.renyi.game(50,0.5)
}
d <- list(a,b)
k <- fraiman(d, printResult = TRUE)

## test under H1
a <- b <- d <- list()
for(i in 1:10){
  a[[i]] <- erdos.renyi.game(50,0.5)
  b[[i]] <- erdos.renyi.game(50,0.6)
}
d <- list(a,b)
k <- fraiman(d, printResult = TRUE)

## End(Not run)
```

---

gCEM *Clustering Expectation-Maximization for Graphs (gCEM)*

Description

gCEM clusters graphs following an expectation-maximization algorithm based on the Kullback-Leibler divergence between the spectral densities of the graph and of the random graph model.

Usage

```
gCEM(g, model, num_clusters, max_iter = 10, ncores = 1)
```

Arguments

g	a list containing the adjacency matrix of the graphs to be clustered.
model	a string that indicates one of the following random graph models: "ER" (Erdos-Renyi random graph), "GRG" (geometric random graph), "KR" (k regular graph), "WS" (Watts-Strogatz model), and "BA" (Barabasi-Albert model).
num_clusters	an integer specifying the number of clusters.
max_iter	the maximum number of expectation-maximization steps to execute.
ncores	the number of cores to be used for the parallel processing. The default value is 1.

Value

a list containing three fields: labels a vector of the same lenght of g containing the clusterization labels; p a vector of length equals to num_clusters;

References

Celeux, Gilles, and Gerard Govaert. "Gaussian parsimonious clustering models." Pattern recognition 28.5 (1995): 781-793.

Examples

```
require(igraph)
 g <- list()
 for(i in 1:2){
   g[[i]] <- igraph::get.adjacency(igraph::sample_gnp(n=10, p=0.5))
 }
 for(i in 3:4){
   g[[i]] <- igraph::get.adjacency(igraph::sample_gnp(n=10, p=1))
 }
 res <- gCEM(g, model="ER", num_clusters=2, max_iter=1, ncores=1)
```

`ghoshdastidar` *Ghoshdastidar hypothesis testing for large random graphs.*

Description

Given two lists of graphs generated by the inhomogeneous random graph model, `ghoshdastidar` tests if they were generated by the same parameters.

Usage

```
ghoshdastidar(
  x,
  y,
  maxPer = 300,
  alpha = 0.05,
  two.sample = FALSE,
  printResult = FALSE
)
```

Arguments

`x`	the first list of undirected graphs to be compared. Must be a list of matrices or igraph objects.
`y`	the second list of undirected graphs to be compared. Must be a list of matrices or igraph objects.
`maxPer`	integer indicating the number of bootstrap resamples (default is 300).

alpha	the significance level for the test (default is 0.05).
two.sample	logical. If TRUE the sets contain only one graph each. If FALSE the sets contain more than one graph each (default is FALSE).
printResult	logical indicating if the test must print the result (default is FALSE).

Value

A list containing:

test_stats	the value of the test.
p_value	the p-value of the test (only returned when the parameter 'two.sample' is FALSE).
bootstrap_samples	The test distrition on the bootstrap resamples (only returned when the parameter 'two.sample' is FALSE).

References

Ghoshdastidar, Debarghya, et al. "Two-sample tests for large random graphs using network statistics". arXiv preprint arXiv:1705.06168 (2017).

Ghoshdastidar, Debarghya, et al. "Two-sample hypothesis testing for inhomogeneous random graphs". arXiv preprint, arXiv:1707.00833 (2017).

Examples

```
## Not run:
require(igraph)
set.seed(42)

## test for sets with more than one graph each under H0
x <- y <- list()
for(i in 1:10){
  x[[i]] <- as.matrix(get.adjacency(erdos.renyi.game(50,0.6)))
  y[[i]] <- as.matrix(get.adjacency(erdos.renyi.game(50,0.6)))
}
D <- ghoshdastidar(x, y, printResult = TRUE)

## test for sets with more than one graph each under H1
x <- y <- list()
for(i in 1:10){
  x[[i]] <- as.matrix(get.adjacency(erdos.renyi.game(50,0.6)))
  y[[i]] <- as.matrix(get.adjacency(erdos.renyi.game(50,0.7)))
}
D <- ghoshdastidar(x, y, printResult = TRUE)

## test for sets with only one graph each under H0
x <- y <- list()
x[[1]] <- erdos.renyi.game(300, 0.6)
y[[1]] <- erdos.renyi.game(300, 0.6)
D <- ghoshdastidar(x, y, two.sample= TRUE, printResult = TRUE)
```

```
## test for sets with only one graph each under H1
x <- y <- list()
x[[1]] <- erdos.renyi.game(300, 0.6)
y[[1]] <- erdos.renyi.game(300, 0.7)
D <- ghoshdastidar(x, y, two.sample= TRUE, printResult = TRUE)

## End(Not run)
```

GIC *Graph Information Criterion (GIC)*

Description

GIC returns the Kullback-Leibler divergence or L2 distance between an undirected graph and a given graph model.

Usage

```
GIC(
  A,
  model,
  p = NULL,
  bandwidth = "Silverman",
  eigenvalues = NULL,
  dist = "KL"
)
```

Arguments

A	the adjacency matrix of the graph. For an unweighted graph it contains only 0s and 1s. For a weighted graph, it may contain nonnegative real values that correspond to the weights of the edges.
model	either a list, a string, a function or a matrix describing a graph model: A list that represents the spectral density of a model. It contains the components "x" and "y", which are numeric vectors of the same size. The "x" component contains the points at which the density was computed and the "y" component contains the observed density. A string that indicates one of the following models: "ER" (Erdos-Renyi random graph), "GRG" (geometric random graph), "KR" (k regular random graph), "WS" (Watts-Strogatz model), and "BA" (Barabasi-Albert model). When the argument 'model' is a string, the user must also provides the model parameter by using the argument 'p'. A function that returns a graph (represented by its adjacency matrix) generated by a graph model. It must contain two arguments: the first one corresponds to the graph size and the second to the parameter of the model. The model parameter will be provided by the argument 'p' of the 'GIC' function.

A matrix containing the spectrum of the model. Each column contains the eigenvalues of a graph generated by the model. To estimate the spectral density of the model, the method will estimate the density of the values of each column, and then will take the average density.

`p` the model parameter. The user must provide a valid parameter if the argument 'model' is a string or a function. For the predefined models ("ER", "GRG", "KR", "WS", and "BA"), the parameter the probability to connect a pair of vertices, for the "ER" model (Erdos-Renyi random graph);
the radius used to construct the geometric graph in a unit square, for the "GRG" model (geometric random graph);
the degree 'k' of a regular graph, for the "KR" model (k regular random graph);
the probability to reconnect a vertex, for the "WS" model (Watts-Strogatz model);
and the scaling exponent, for the "BA" model (Barabasi-Albert model).

`bandwidth` string showing which criterion is used to choose the bandwidth during the spectral density estimation. Choose between the following criteria: "Silverman" (default), "Sturges" and "bcv". "bcv" is an abbreviation of biased cross-validation.

`eigenvalues` optional parameter. It contains the eigenvalues of matrix A. Then, it can be used when the eigenvalues of A were previously computed. If no value is passed, then the eigenvalues of A will be computed by 'GIC'.

`dist` string indicating if you want to use the "KL" (default) or "L2" distances. "KL" means Kullback-Leibler divergence.

Value

A real number corresponding to the Kullback-Leibler divergence or L2 distance between the observed graph and the graph model.

References

Takahashi, D. Y., Sato, J. R., Ferreira, C. E. and Fujita A. (2012) Discriminating Different Classes of Biological Networks by Analyzing the Graph Spectra Distribution. _PLoS ONE_, *7*, e49949. doi:10.1371/journal.pone.0049949.

Silverman, B. W. (1986) _Density Estimation_. London: Chapman and Hall.

Sturges, H. A. The Choice of a Class Interval. _J. Am. Statist. Assoc._, *21*, 65-66.

Examples

```
A <- as.matrix(igraph::get.adjacency(igraph::sample_gnp(n=50, p=0.5)))
# Using a string to indicate the graph model
result1 <- GIC(A, "ER", 0.5)
result1

# Using a function to describe the graph model
# Erdos-Renyi graph
model <- function(n, p) {
   return(as.matrix(igraph::get.adjacency(igraph::sample_gnp(n, p))))
}
result2 <- GIC(A, model, 0.5)
```

```
result2
```

`graph.acf` *Auto Correlation Function Estimation for Graphs*

Description

The function `graph.acf` computes estimates of the autocorrelation function for graphs.

Usage

```
graph.acf(x, plot = TRUE)
```

Arguments

`x`	a list of adjacency (symmetric) matrices of undirected graphs. For unweighted graphs, each matrix contains only 0s and 1s. For weighted graphs, each matrix may contains real values that correspond to the weights of the edges.
`plot`	logical. If TRUE (default) the graph.acf is plotted.

Value

An object of class acf.

References

Fujita, A., Takahashi, D. Y., Balardin, J. B., Vidal, M. C. and Sato, J. R. (2017) Correlation between graphs with an application to brain network analysis. _Computational Statistics & Data Analysis_ *109*, 76-92.

Examples

```
require(igraph)
x <- list()
p <- array(0, 100)
p[1:3] <- rnorm(3)
for (t in 4:100) {
    p[t] <- 0.5*p[t-3] + rnorm(1)
}
ma <- max(p)
mi <- min(p)
p <- (p - mi)/(ma-mi)
for (t in 1:100) {
    x[[t]] <- get.adjacency(erdos.renyi.game(100, p[t]))
}
graph.acf(x, plot=TRUE)
```

`graph.cluster` *Hierarchical cluster analysis on a list of graphs.*

Description

Given a list of graphs, `graph.cluster` builds a hierarchy of clusters according to the Jensen-Shannon divergence between graphs.

Usage

```
graph.cluster(x, k, method = "complete", bandwidth = "Silverman")
```

Arguments

`x`	a list of adjacency (symmetric) matrices of undirected graphs. For unweighted graphs, each matrix contains only 0s and 1s. For weighted graphs, each matrix may contains real values that correspond to the weights of the edges.
`k`	the number of clusters.
`method`	the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of '"ward.D"', '"ward.D2"', '"single"', '"complete"', '"average"' (= UPGMA), '"mcquitty"' (= WPGMA), '"median"' (= WPGMC) or '"centroid"' (= UPGMC).
`bandwidth`	string indicating which criterion will be used to choose the bandwidth for the spectral density estimation. The available criteria are "Silverman" (default) and "Sturges".

Value

A list containing:

`hclust`	an object of class *hclust* which describes the tree produced by the clustering process.
`cluster`	the clustering labels for each graph.

References

Takahashi, D. Y., Sato, J. R., Ferreira, C. E. and Fujita A. (2012) Discriminating Different Classes of Biological Networks by Analyzing the Graph Spectra Distribution. _PLoS ONE_, *7*, e49949. doi:10.1371/journal.pone.0049949.

Silverman, B. W. (1986) _Density Estimation_. London: Chapman and Hall.

Sturges, H. A. The Choice of a Class Interval. _J. Am. Statist. Assoc._, *21*, 65-66.

Examples

```
require(igraph)
g <- list()
for (i in 1:5) {
    g[[i]] <- as.matrix(get.adjacency(
                        erdos.renyi.game(50, 0.5, type="gnp",
                                         directed = FALSE)))
}
for (i in 6:10) {
    g[[i]] <- as.matrix(get.adjacency(
                        watts.strogatz.game(1, 50, 8, 0.2)))
}
for (i in 11:15) {
    g[[i]] <- as.matrix(get.adjacency(
                        barabasi.game(50, power = 1,
                                      directed = FALSE)))
}
graph.cluster(g, 3)
```

---

graph.cor.test *Test for Association / Correlation Between Paired Samples of Graphs*

Description

graph.cor.test tests for association between paired samples of graphs, using Spearman's rho correlation coefficient.

Usage

```
graph.cor.test(x, y)
```

Arguments

x	a list of adjacency (symmetric) matrices of undirected graphs. For unweighted graphs, each matrix contains only 0s and 1s. For weighted graphs, each matrix contains real values that correspond to the weights of the edges.
y	a list with the same length of 'x'. It contains adjacency (symmetric) matrices of undirected graphs. For unweighted graphs, each matrix contains only 0s and 1s. For weighted graphs, each matrix may contain real values that correspond to the weights of the edges.

Value

statistic	the value of the test statistic.
p.value	the p-value of the test.
estimate	the estimated measure of association 'rho'.

References

Fujita, A., Takahashi, D. Y., Balardin, J. B., Vidal, M. C. and Sato, J. R. (2017) Correlation between graphs with an application to brain network analysis. _Computational Statistics & Data Analysis_ *109*, 76-92.

Examples

```
require(igraph)
x <- list()
y <- list()

p <- MASS::mvrnorm(50, mu=c(0,0), Sigma=matrix(c(1, 0.5, 0.5, 1), 2, 2))

ma <- max(p)
mi <- min(p)
p[,1] <- (p[,1] - mi)/(ma - mi)
p[,2] <- (p[,2] - mi)/(ma - mi)

for (i in 1:50) {
    x[[i]] <- get.adjacency(erdos.renyi.game(50, p[i,1]))
    y[[i]] <- get.adjacency(erdos.renyi.game(50, p[i,2]))
}

graph.cor.test(x, y)
```

---

graph.entropy *Graph spectral entropy*

Description

graph.entropy returns the spectral entropy of an undirected graph.

Usage

```
graph.entropy(A = NULL, bandwidth = "Silverman", eigenvalues = NULL)
```

Arguments

A	the adjacency matrix of the graph. For an unweighted graph, it contains only 0s and 1s. For a weighted graph, it may contain nonnegative real numbers that correspond to the weights of the edges.
bandwidth	string showing which criterion is used to choose the bandwidth during the spectral density estimation. Choose between the following criteria: "Silverman" (default), "Sturges" and "bcv". "bcv" is an abbreviation of biased cross-validation.
eigenvalues	optional parameter. It contains the eigenvalues of matrix A. Then, if the eigenvalues of matrix A have already been computed, this parameter can be used instead of A. If no value is passed, then the eigenvalues of A will be computed by 'graph.entropy'.

Value

a real number corresponding to the graph spectral entropy.

References

Takahashi, D. Y., Sato, J. R., Ferreira, C. E. and Fujita A. (2012) Discriminating Different Classes of Biological Networks by Analyzing the Graph Spectra Distribution. _PLoS ONE_, *7*, e49949. doi:10.1371/journal.pone.0049949.

Silverman, B. W. (1986) _Density Estimation_. London: Chapman and Hall.

Sturges, H. A. The Choice of a Class Interval. _J. Am. Statist. Assoc._, *21*, 65-66.

Examples

```
G <- igraph::sample_gnp(n=100, p=0.5)
A <- as.matrix(igraph::get.adjacency(G))
entropy <- graph.entropy(A)
entropy
```

---

graph.model.selection *Graph model selection*

Description

graph.model.selection selects the graph model that best approximates the observed graph according to the Graph Information Criterion (GIC).

Usage

```
graph.model.selection(
  A,
  models = NULL,
  parameters = NULL,
  eps = 0.01,
  bandwidth = "Silverman",
  eigenvalues = NULL
)
```

Arguments

A	the adjacency (symmetric) matrix of an undirected graph. For an unweighted graph it contains only 0s and 1s. For a weighted graph, it contains real values that correspond to the weights of the edges.

`models` either a vector of strings, a list of functions or a list of arrays describing graph models:

A vector of strings containing some of the following models: "ER" (Erdos-Renyi random graph), "GRG" (geometric random graph), "KR" (k regular random graph), "WS" (Watts-Strogatz model), and "BA" (Barabasi-Albert model).

A list of functions. Each function returns a graph (represented by its adjacency matrix) generated by a graph model and has two arguments: the graph size and the model parameter, in this order.

A list of arrays. Each elememt of the list is a three-dimensional array containing the precomputed spectrum of each model. Let M be a graph model. For each parameter p considered for M, the array of model M contains the eigenvalues of graphs randomly generated by M with parameter p. The position (i,j,k) of the array contains the j-th eigenvalue of the k-th graph that generated by M with the i-th parameter. The attribute 'rownames' of the array corresponds to the parameters converted to string.

If the argument "models" is NULL, then the "ER", "WS", and "BA" models will be considered for the model selection.

`parameters` list of numeric vectors. Each vector contains the values that will be considerated for the parameter estimation of the corresponding model. If the user does not provide the argument 'parameters', then default values are used for the predefined models ("ER", "GRG", "KR", "WS", and "BA"). The default vector corresponds to a sequence from

0 to 1 with step 'eps' for the "ER" model (Erdos-Renyi random graph), in which the parameter corresponds to the probability to connect a pair of vertices;

0 to sqrt(2) with step 'eps' for the "GRG" model (geometric random graph), in which the parameter corresponds to the radius used to contruct the geometric graph in a unit square;

0 to 'n' with step 'n*eps' for the "KR" model (k regular random graph), in which the parameter of the model corresponds to the degree 'k' of a regular graph;

0 to 1 with step 'eps' for the "WS" model (Watts-Strogatz model), in which the parameter corresponds to the probability to reconnect a vertex;

and 0 to 3 with step 'eps' for the "BA" model (Barabasi-Albert model), in which the parameter corresponds to the scaling exponent.

`eps` precision of the grid (default is 0.01).

`bandwidth` string indicating which criterion will be used to choose the bandwidth for the spectral density estimation. The available criteria are "Silverman" (default) and "Sturges".

`eigenvalues` optional parameter. It contains the eigenvalues of matrix A. Then, it can be used when the eigenvalues of A were previously computed. If no value is passed, then the eigenvalues of A will be computed by 'graph.model.selection'.

Value

A list containing:

`model` the indice(s) or name(s) of the selected model(s), i. e. the model(s) that minimize(s) the Graph Information Criterion (GIC).

estimates — a matrix in which each row corresponds to a model, the column "p" corresponds to the parameter estimate, and the column "GIC" corresponds to the Graph Information Criterion (GIC), i. e. the Kullback-Leibler divergence between the observed graph and the model.

References

Takahashi, D. Y., Sato, J. R., Ferreira, C. E. and Fujita A. (2012) Discriminating Different Classes of Biological Networks by Analyzing the Graph Spectra Distribution. _PLoS ONE_, *7*, e49949. doi:10.1371/journal.pone.0049949.

Silverman, B. W. (1986) _Density Estimation_. London: Chapman and Hall.

Sturges, H. A. The Choice of a Class Interval. _J. Am. Statist. Assoc._, *21*, 65-66.

Examples

```
require(igraph)
A <- as.matrix(get.adjacency(erdos.renyi.game(30, p=0.5)))
# Using strings to indicate the graph models
result1 <- graph.model.selection(A, models=c("ER", "WS"), eps=0.5)
result1
# Using functions to describe the graph models
# Erdos-Renyi graph
model1 <- function(n, p) {
   return(as.matrix(get.adjacency(erdos.renyi.game(n, p))))
}
# Watts-Strougatz graph
model2 <- function(n, pr, K=8) {
    return(as.matrix(get.adjacency(watts.strogatz.game(1, n, K, pr))))
}
parameters <- list(seq(0, 1, 0.5), seq(0, 1, 0.5))
result2 <- graph.model.selection(A, list(model1, model2), parameters)
result2
```

---

graph.mult.scaling *Multidimensional scaling of graphs*

Description

graph.mult.scaling performs multidimensional scaling of graphs. It takes the Jensen-Shannon divergence between graphs (JS) and uses the 'cmdscale' function from the 'stats' package to obtain a set of points such that the distances between the points are similar to JS.

Usage

```
graph.mult.scaling(
  x,
  plot = TRUE,
```

```
  bandwidth = "Silverman",
  type = "n",
  main = "",
  ...
)
```

Arguments

x	a list of adjacency (symmetric) matrices of undirected graphs. For unweighted graphs, each matrix contains only 0s and 1s. For weighted graphs, each matrix may contains real values that correspond to the weights of the edges.
plot	logical. If TRUE (default) the points chosen to represent the Jensen-Shannon divergence between graphs are plotted.
bandwidth	character string indicating which criterion will be used to choose the bandwidth for the spectral density estimation. The available criteria are "Silverman" (default) and "Sturges".
type	what type of plot should be drawn. The defaut value is '"n"', which indicates that the points will not be plotted (i. e. only the labels of the graphs will be plotted).
main	title of the plot (default value is "").
...	additional plotting parameters. See 'plot' function from the 'graphics' package for the complete list.

Value

A matrix in which each column corresponds to a coordinate and each row corresponds to a graph (point). Then, each row gives the coordinates of the points chosen to represent the Jensen-Shannon divergence between graphs.

References

Takahashi, D. Y., Sato, J. R., Ferreira, C. E. and Fujita A. (2012) Discriminating Different Classes of Biological Networks by Analyzing the Graph Spectra Distribution. _PLoS ONE_, *7*, e49949. doi:10.1371/journal.pone.0049949.

Silverman, B. W. (1986) _Density Estimation_. London: Chapman and Hall.

Sturges, H. A. The Choice of a Class Interval. _J. Am. Statist. Assoc._, *21*, 65-66.

Examples

```
require(igraph)
g <- list()
for (i in 1:5) {
    g[[i]] <- as.matrix(get.adjacency(
                        erdos.renyi.game(50, 0.5, type="gnp",
                                         directed = FALSE)))
}
for (i in 6:10) {
    g[[i]] <- as.matrix(get.adjacency(
```

```
                        watts.strogatz.game(1, 50, 8, 0.2)))
}
for (i in 11:15) {
    g[[i]] <- as.matrix(get.adjacency(
                        barabasi.game(50, power = 1,
                                      directed = FALSE)))
}
graph.mult.scaling(g)
```

`graph.param.estimator` *Graph parameter estimator*

Description

`graph.param.estimator` estimates the parameter that best approximates the model to the observed graph according to the Graph Information Criterion (GIC).

Usage

```
graph.param.estimator(
  A,
  model,
  parameters = NULL,
  eps = 0.01,
  bandwidth = "Silverman",
  eigenvalues = NULL,
  spectra = NULL,
  classic = FALSE
)
```

Arguments

`A`	the adjacency matrix of the graph. For an unweighted graph it contains only 0s and 1s. For a weighted graph, it may contain nonnegative real values that correspond to the weights of the edges.
`model`	either a string or a function: A string that indicates one of the following models: "ER" (Erdos-Renyi random graph), "GRG" (geometric random graph), "KR" (k regular random graph), "WS" (Watts-Strogatz model), and "BA" (Barabasi-Albert model). A function that returns a graph (represented by its adjacency matrix) generated by a graph model. It must contain two arguments: the first one corresponds to the graph size and the second to the parameter of the model.
`parameters`	numeric vector containing the values that that will be considered for the parameter estimation. The 'graph.param.estimator' will return the element of 'parameter' that minimizes the Graph Information Criterion (GIC). If the user does not

provide the argument 'parameters', and 'model' is an array, then the values considered for the parameter estimation are the rownames converted to a numeric vector. If 'model' is a string, then default values are used for the predefined models ("ER", "GRG", "KR", "WS", and "BA"). The default 'parameter' argument corresponds to a sequence from

0 to 1 with step 'eps' for the "ER" model (Erdos-Renyi random graph), in which the parameter corresponds to the probability to connect a pair of vertices;

0 to sqrt(2) with step 'eps' for the "GRG" model (geometric random graph), in which the parameter corresponds to the radius used to construct the geometric graph in a unit square;

0 to 'n' with step 'n*eps' for the "KR" model (k regular random graph), in which the parameter of the model corresponds to the degree 'k' of a regular graph;

0 to 1 with step 'eps' for the "WS" model (Watts-Strogatz model), in which the parameter corresponds to the probability to reconnect a vertex;

and 0 to 3 with step 'eps' for the "BA" model (Barabasi-Albert model), in which the parameter corresponds to the scaling exponent.

`eps` precision of the grid (default is 0.01) when 'classic' is TRUE.

`bandwidth` string indicating which criterion will be used to choose the bandwidth for the spectral density estimation. The available criteria are "Silverman" (default) and "Sturges".

`eigenvalues` optional parameter. It contains the eigenvalues of matrix A. Then, it can be used when the eigenvalues of A were previously computed. If no value is passed, then the eigenvalues of A will be computed by 'graph.param.estimator'.

`spectra` optional parameter containing the precomputed spectrum of the model. It is a three-dimensional array in which the first dimension corresponds to all parameters that will be explored in the grid, the second dimension has the same size of the given graph, and the third one corresponds to graphs randomly generated by the model. Thus, the position (i,j,k) contains the j-th eigenvalue of the k-th graph generated with the i-th parameter. The attribute 'rownames' of the array corresponds to the parameters converted to string. If spectra is NULL (default), then 'model' is used to generate random graphs and their spectra are computed automatically.

`classic` logical. If FALSE (default) parameter is estimated using ternary search. If TRUE parameter is estimated using grid search.

Value

A list containing:

`p` the parameter estimate. For the "ER", "GRG", "KR", "WS", and "BA" models, the parameter corresponds to the probability to connect a pair of vertices, the radius used to construct the geometric graph in a unit square, the degree k of a regular graph, the probability to reconnect a vertex, and the scaling exponent, respectively.

`KL` the Graph Information Criterion (GIC), i. e. the Kullback-Leibler divergence between the observed graph and the graph model with the estimated parameter.

References

Takahashi, D. Y., Sato, J. R., Ferreira, C. E. and Fujita A. (2012) Discriminating Different Classes of Biological Networks by Analyzing the Graph Spectra Distribution. _PLoS ONE_, *7*, e49949. doi:10.1371/journal.pone.0049949.

Silverman, B. W. (1986) _Density Estimation_. London: Chapman and Hall.

Sturges, H. A. The Choice of a Class Interval. _J. Am. Statist. Assoc._, *21*, 65-66.

Examples

```
require(igraph)
A <- as.matrix(get.adjacency(erdos.renyi.game(50, p=0.5)))

# Using a string to indicate the graph model
result1 <- graph.param.estimator(A, "ER", eps=0.25)
result1

## Using a function to describe the graph model
## Erdos-Renyi graph
# model <- function(n, p) {
#    return(as.matrix(get.adjacency(erdos.renyi.game(n, p))))
# }
# result2 <- graph.param.estimator(A, model,  seq(0.2, 0.8, 0.1))
# result2
```

`graph.test` *Test for the Jensen-Shannon divergence between graphs*

Description

`graph.test` tests whether two sets of graphs were generated by the same random graph model. This bootstrap test is based on the Jensen-Shannon (JS) divergence between graphs.

Usage

```
graph.test(x, y, numBoot = 1000, bandwidth = "Silverman")
```

Arguments

`x`	a list of adjacency (symmetric) matrices. For unweighted graphs, each matrix contains only 0s and 1s. For weighted graphs, each matrix contains real values that correspond to the weights of the edges.
`y`	a list of adjacency (symmetric) matrices. For unweighted graphs, each matrix contains only 0s and 1s. For weighted graphs, each matrix contains real values that correspond to the weights of the edges.
`numBoot`	integer indicating the number of bootstrap resamplings.
`bandwidth`	string indicating which criterion will be used to choose the bandwidth for the spectral density estimation. The available criteria are "Silverman" (default) and "Sturges".

Details

Given two lists of graphs, 'x' and 'y', 'graph.test' tests H0: "JS divergence between 'x' and 'y' is 0" against H1: "JS divergence between 'x' and 'y' is larger than 0".

Value

A list containing:

JS	the Jensen-Shannon divergence between 'x' and 'y'.
p.value	the p-value of the test.

References

Takahashi, D. Y., Sato, J. R., Ferreira, C. E. and Fujita A. (2012) Discriminating Different Classes of Biological Networks by Analyzing the Graph Spectra Distribution. _PLoS ONE_, *7*, e49949. doi:10.1371/journal.pone.0049949.

Silverman, B. W. (1986) _Density Estimation_. London: Chapman and Hall.

Sturges, H. A. The Choice of a Class Interval. _J. Am. Statist. Assoc._, *21*, 65-66.

Examples

```
library(igraph)
x <- y <- list()
for (i in 1:20)
   x[[i]] <- as.matrix(get.adjacency(erdos.renyi.game(50, p=0.5)))
for (i in 1:20)
   y[[i]] <- as.matrix(get.adjacency(erdos.renyi.game(50, p=0.51)))

result <- graph.test(x, y, numBoot=100)
result
```

---

kmeans.graph *K-means for Graphs*

Description

kmeans.graph clusters graphs following a k-means algorithm based on the Jensen-Shannon divergence between the spectral densities of the graphs.

Usage

```
kmeans.graph(x, k, nstart = 2)
```

Arguments

x	a list containing the adjacency matrix of the graphs to be clustered.
k	an integer specifying the number of clusters.
nstart	the number of trials of k-means clusterizations. The algorithm returns the clusterization with the best silhouette.

Value

a vector of the same lenght of g containing the clusterization labels.

References

MacQueen, James. "Some methods for classification and analysis of multivariate observations." Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Vol. 1. No. 14. 1967.

Lloyd, Stuart. "Least squares quantization in PCM." IEEE transactions on information theory 28.2 (1982): 129-137.

Examples

```
require(igraph)
g <- list()
for(i in 1:5){
  g[[i]] <- get.adjacency(sample_gnp(30, p=0.2))
}
for(i in 6:10){
  g[[i]] <- get.adjacency(sample_gnp(30, p=0.5))
}
res <- kmeans.graph(g, k=2, nstart=2)
```

---

sp.anogva *Semi-Parametric Analysis Of Graph Variability (ANOGVA)*

Description

sp.anogva statistically tests whether two or more graphs are generated by the same model and set of parameters.

Usage

```
sp.anogva(
  graph,
  model,
  maxBoot = 500,
  spectra = NULL,
  eps = 0.01,
  classic = FALSE
)
```

Arguments

graph	a list of adjacency (symmetric) matrices of the undirected graphs to be compared. For unweighted graphs, each matrix contains only 0s and 1s. For weighted graphs, each matrix contains real values that correspond to the weights of the edges.
model	A string that indicates one of the following models: "ER" (Erdos-Renyi random graph model), "GRG" (geometric random graph model), "WS" (Watts-Strogatz random graph model), and "BA" (Barabasi-Albert random graph model).
maxBoot	integer indicating the number of bootstrap resamples (default is 500).
spectra	optional parameter containing the precomputed spectrum of the model. It is a three-dimensional array in which the first dimension corresponds to all parameters that will be explored in the parameter estimation, the second dimension has the same size of the given graph, and the third one corresponds to graphs randomly generated by the model. Thus, the position (i,j,k) contains the j-th eigenvalue of the k-th graph generated with the i-th parameter. The attribute 'rownames' of the array corresponds to the parameters converted to string. If spectra is NULL (default), then model' is used to generate random graphs and their spectra are computed automatically.
eps	(default is 0.01) precision of the grid when 'classic' = TRUE.
classic	logical. If FALSE (default) parameter is estimated using ternary search, if TRUE parameter is estimated using grid search.

Value

A list containing:

parameter	an array containing the estimated parameters for each graph.
F.value	the F statistic of the test.
p.value	the p-value of the test.

References

Andre Fujita, Eduardo Silva Lira, Suzana de Siqueira Santos, Silvia Yumi Bando, Gabriela Eleuterio Soares, Daniel Yasumasa Takahashi. A semi-parametric statistical test to compare complex networks, Journal of Complex Networks, cnz028, https://doi.org/10.1093/comnet/cnz028

Examples

```
## Please uncomment the following lines to run an example
# require(igraph)
# set.seed(42)
# model <- "ER"
# graph <- list()

## Under H0
# graph[[1]] <- get.adjacency(erdos.renyi.game(50, 0.5))
# graph[[2]] <- get.adjacency(erdos.renyi.game(50, 0.5))
```

```
# graph[[3]] <- get.adjacency(erdos.renyi.game(50, 0.5))
# result <- sp.anogva(graph, model, maxBoot = 300)
# result

## Under H1
# graph[[1]] <- get.adjacency(erdos.renyi.game(50, 0.5))
# graph[[2]] <- get.adjacency(erdos.renyi.game(50, 0.55))
# graph[[3]] <- get.adjacency(erdos.renyi.game(50, 0.5))
# result <- sp.anogva(graph, model, maxBoot = 300)
# result
```

---

tang *Tang hypothesis testing for random graphs.*

Description

Given two independent finite-dimensional random dot product graphs, tang tests if they have generating latent positions that are drawn from the same distribution.

Usage

```
tang(
  G1,
  G2,
  dim,
  sigma = NULL,
  alpha = 0.05,
  bootstrap_sample = 200,
  printResult = FALSE
)
```

Arguments

G1	the first undirected graph to be compared. Must be an igraph object.
G2	the second undirected graph to be compared. Must be an igraph object.
dim	dimension of the adjacency spectral embedding.
sigma	a real value indicating the kernel bandwidth. If NULL (default) the bandwidth is calculated by the method.
alpha	the significance level for the test (default is 0.05).
bootstrap_sample	integer indicating the number of bootstrap resamples (default is 200).
printResult	logical indicating if the test must print the result (default is FALSE).

Value

A list containing:

X1	the embedding of G1.
X2	the embedding of G2.
test_stats	the value of the test.
p_value	the p-value of the test.
bootstrap_samples	The test distrition on the bootstrap resamples.

References

Tang, Minh, et al. "A nonparametric two-sample hypothesis testing problem for random graphs." Bernoulli 23.3 (2017): 1599-1630.

Tang, Minh, et al. "A semiparametric two-sample hypothesis testing problem for random graphs." Journal of Computational and Graphical Statistics 26.2 (2017): 344-354.

Examples

```
require(igraph)
set.seed(42)

## test under H0
lpvs <- matrix(rnorm(200), 20, 10)
lpvs <- apply(lpvs, 2, function(x) { return (abs(x)/sqrt(sum(x^2))) })
g1 <- sample_dot_product(lpvs)
g2 <- sample_dot_product(lpvs)
D <- tang(g1,g2, 5, printResult = TRUE)

## test under H1
lpvs2 <- matrix(pnorm(200), 20, 10)
lpvs2 <- apply(lpvs2, 2, function(x) { return (abs(x)/sqrt(sum(x^2))) })
g2 <- suppressWarnings(sample_dot_product(lpvs2))
D <- tang(g1,g2, 5, printResult = TRUE)
```

Index