Package 'statGraph'

December 21, 2022

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]
Florencia G. Leonardi [ctb],
Andressa Cerqueira [ctb]

Maintainer Andre Fujita <andrefujita@usp.br>

Depends R (>= 3.6.0), stats, graphics

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

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 0.5.1

Date 2021-08-14

RoxygenNote 7.2.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-08-14 21:40:10 UTC

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

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:	0.5.1
Date:	2021-08-14
Depends:	R (>= 3.6.0), stats, graphics
Imports:	igraph, MASS, rARPACK, cluster, foreach, parallel, doParallel, methods
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] Florencia G. Leonardi [ctb], Andressa Cerqueira [ctb]

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 'takahashi.test' function.

Usage

```
anogva(G, labels, maxBoot = 1000, bandwidth = "Silverman")
```

Arguments

G	a list of undirected graphs (igraph type) or their adjacency matrices. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative real values that correspond to the weights of the edges.
labels	an array of integers indicating the labels of each graph.
maxBoot	integer indicating the number of bootstrap resamplings.
bandwidth	string showing which criterion is used to choose the bandwidth during the spectral density estimation. Choose between the following criteria: "Silverman" (default), "Sturges", "bcv", "ucv" and "SJ". "bcv" is an abbreviation of biased cross-validation, while "ucv" means unbiased cross-validation. "SJ" implements the methods of Sheather & Jones (1991) to select the bandwidth using pilot estimation of derivatives.

Value

A list with class "htest" containing the following components:

statistic	the statistic of the test.
p.value	the p-value of the test.
method	a string indicating the used method.
data.name	a string with the data's name(s).

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.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. _Journal of the Royal Statistical Society series B_, 53, 683-690. http://www.jstor.org/stable/2345597.

Examples

```
set.seed(1)
g1 <- g2 <- g3 <- list()
for (i in 1:20){
  g1[[i]] <- igraph::sample_gnp(50, 0.50)
  g2[[i]] <- igraph::sample_gnp(50, 0.50)
```

```
  g3[[i]] <- igraph::sample_gnp(50, 0.52)
}
G <- c(g1, g2, g3)
label <- c(rep(1, 20), rep(2, 20), rep(3, 20))
result <- anogva(G, label, maxBoot=50)
result
```

`cerqueira.test` *Cerqueira et al.'s test*

Description

Given two identically independently distributed (idd) samples of graphs G1 and G2, the test verifies if they have the same distribution by calculating the mean distance D from G1 to G2. 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.test(G1, G2, maxBoot = 300)
```

Arguments

`G1`	the first iid sample of graphs to be compared. Must be a list of igraph objects.
`G2`	the second iid sample of graphs to be compared. Must be a list of igraph objects.
`maxBoot`	integer indicating the number of bootstrap resamples (default is 300).

Value

A list with class "htest" containing the following components:

`statistic`	the W-value of the test.
`p.value`	the p-value of the test.
`method`	a string indicating the used method.
`data.name`	a string with the data's name(s).

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:
set.seed(42)

## test under H0
G1 <- G2 <- list()
for(i in 1:10){
  G1[[i]] <- igraph::sample_gnp(50, 0.5)
  G2[[i]] <- igraph::sample_gnp(50, 0.5)
}
k1 <- cerqueira.test(G1, G2)
k1

## test under H1
G1 <- G2 <- list()
for(i in 1:10){
  G1[[i]] <- igraph::sample_gnp(50, 0.5)
  G2[[i]] <- igraph::sample_gnp(50, 0.6)
}
k2 <- cerqueira.test(G1, G2)
k2

## End(Not run)
```

---

eigenvalue.probability

Degree-based eigenvalue probability

Description

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

Usage

```
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

```
set.seed(42)
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(eigenvalue.probability(deg_prob, q_prob, all_k, z))
eigenval_prob
```

`fraiman.test` *Fraiman's test*

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.test(G, maxBoot = 300)
```

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.
maxBoot	integer indicating the number of bootstrap resamples (default is 300).

Value

A list with class "htest" containing the following components:

statistic	the T-value of the test.
p.value	the p-value of the test.
method	a string indicating the used method.
data.name	a string with the data's name(s).

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:
set.seed(42)

## test under H0
a <- b <- G <- list()
for(i in 1:10){
  a[[i]] <- igraph::sample_gnp(50, 0.5)
  b[[i]] <- igraph::sample_gnp(50, 0.5)
}
G <- list(a,b)
k1 <- fraiman.test(G)
k1

## test under H1
a <- b <- G <- list()
for(i in 1:10){
  a[[i]] <- igraph::sample_gnp(50, 0.5)
  b[[i]] <- igraph::sample_gnp(50, 0.6)
}
G <- list(a,b)
k2 <- fraiman.test(G)
k2

## End(Not run)
```

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

Description

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

Usage

```
get.spectral.density(G, from = NULL, to = NULL, npoints = 2000, numCores = 1)
```

Arguments

`G`	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

A list with class "statGraph" containing the following components:

`method`	a string indicating the used method.
`info`	a string showing details about the method.
`data.name`	a string with the data's name(s).
`x`	x-value of the the degree-based spectral density of the graph.
`y`	y-value of the the degree-based spectral density of the graph.

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

```
set.seed(42)
G <- igraph::sample_smallworld(dim=1, size=100, nei=2, p=0.2)

# Obtain the degree-based spectral density
density_ <- get.spectral.density(G=G, npoints=80, numCores=1)
density_
```

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

Description

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

Usage

```
ghoshdastidar.test(G1, G2, maxBoot = 300, two.sample = FALSE)
```

Arguments

`G1`	the first list of undirected graphs to be compared. Must be a list of matrices or igraph objects.
`G2`	the second list of undirected graphs to be compared. Must be a list of matrices or igraph objects.
`maxBoot`	integer indicating the number of bootstrap resamples (default is 300).
`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).

Value

A list with class "htest" containing the following components:

`statistic`	the T-value of the test.
`p.value`	the p-value of the test (only returned when the parameter 'two.sample' is FALSE).
`method`	a string indicating the used method.
`data.name`	a string with the data's name(s).

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:
set.seed(42)

## test for sets with more than one graph each under H0
G1 <- G2 <- list()
for(i in 1:10){
  G1[[i]] <- as.matrix(igraph::get.adjacency(igraph::sample_gnp(50,0.6)))
  G2[[i]] <- as.matrix(igraph::get.adjacency(igraph::sample_gnp(50,0.6)))
}
D1 <- ghoshdastidar.test(G1, G2)
D1

## test for sets with more than one graph each under H1
G1 <- G2 <- list()
for(i in 1:10){
  G1[[i]] <- as.matrix(igraph::get.adjacency(igraph::sample_gnp(50,0.6)))
  G2[[i]] <- as.matrix(igraph::get.adjacency(igraph::sample_gnp(50,0.7)))
}
D2 <- ghoshdastidar.test(G1, G2)
D2

## test for sets with only one graph each under H0
G1 <- G2 <- list()
G1[[1]] <- igraph::sample_gnp(300, 0.6)
G2[[1]] <- igraph::sample_gnp(300, 0.6)
D3 <- ghoshdastidar.test(G1, G2, two.sample= TRUE)
D3

## test for sets with only one graph each under H1
G1 <- G2 <- list()
G1[[1]] <- igraph::sample_gnp(300, 0.6)
G2[[1]] <- igraph::sample_gnp(300, 0.7)
D4 <- ghoshdastidar.test(G1, G2, two.sample= TRUE)
D4

## 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(
  G,
  model,
  p = NULL,
  bandwidth = "Silverman",
  eigenvalues = NULL,
  dist = "KL"
)
```

Arguments

G	the undirected graph (igraph type) or its adjacency matrix. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have 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", "bcv", "ucv" and "SJ". "bcv" is an abbreviation of biased cross-validation, while "ucv" means unbiased cross-validation. "SJ" implements

the methods of Sheather & Jones (1991) to select the bandwidth using pilot estimation of derivatives.

`eigenvalues` optional parameter. It contains the eigenvalues of matrix G. Then, it can be used when the eigenvalues of G were previously computed. If no value is passed, then the eigenvalues of G 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 list with class "statGraph" containing the following components:

`method` a string indicating the used method.

`info` a string showing details about the method.

`data.name` a string with the data's name(s).

`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.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. _Journal of the Royal Statistical Society series B_, 53, 683-690. http://www.jstor.org/stable/2345597.

Examples

```
set.seed(1)
G <- igraph::sample_gnp(n=50, p=0.5)

# Using a string to indicate the graph model
result1 <- GIC(G, "ER", 0.5)
result1

# Using a function to describe the graph model
# Erdos-Renyi graph
model <- function(n, p){
   return(igraph::sample_gnp(n, p))
}
result2 <- GIC(G, 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(G, plot = TRUE)
```

Arguments

G	a list of undirected graphs (igraph type) or their adjacency matrices. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative 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

```
set.seed(1)
G <- 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){
  G[[t]] <- igraph::sample_gnp(100, p[t])
}
graph.acf(G, plot=TRUE)
```

graph.cem *Clustering Expectation-Maximization for Graphs (graph.cem)*

Description

graph.cem 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

```
graph.cem(
  g,
  model,
  k,
  max_iter = 10,
  ncores = 1,
  bandwidth = "Sturges",
  parameters = NULL
)
```

Arguments

g	a list containing the graphs or their adjacency matrices 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).
k	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.
bandwidth	string showing which criterion is used to choose the bandwidth during the spectral density estimation. Choose between the following criteria: "Silverman" (default), "Sturges", "bcv", "ucv" and "SJ". "bcv" is an abbreviation of biased cross-validation, while "ucv" means unbiased cross-validation. "SJ" implements the methods of Sheather & Jones (1991) to select the bandwidth using pilot estimation of derivatives.
parameters	a list with the range where the parameters will be estimated. If nothing is passed default values are used for each model.

Value

A list with class "statGraph" containing the following components:

method	a string indicating the used method.
info	a string showing details about the method.

data.name	a string with the data's name(s).
cluster	a vector of the same length of g containing the clusterization labels.
parameters	a vector containing the estimated parameters for the groups. It has the length equals to k.

References

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

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. _Journal of the Royal Statistical Society series B_, 53, 683-690. http://www.jstor.org/stable/2345597.

Examples

```
set.seed(42)
g <- list()
for(i in 1:2){
  g[[i]] <- igraph::sample_gnp(n=10, p=0.5)
}
for(i in 3:4){
  g[[i]] <- igraph::sample_gnp(n=10, p=1)
}
res <- graph.cem(g, model="ER", k=2, max_iter=1, ncores=1)
res
```

---

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(G1, G2)
```

Arguments

G1	a list of undirected graphs (igraph type) or their adjacency matrices. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative real values that correspond to the weights of the edges.
G2	a list of undirected graphs (igraph type) or their adjacency matrices. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative real values that correspond to the weights of the edges.

Value

A list with class "htest" containing the following components:

statistic	the value of the test statistic.
p.value	the p-value of the test.
method	a string indicating the used method.
data.name	a string with the data's name(s).
estimates	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

```
set.seed(1)
G1 <- G2 <- 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){
  G1[[i]] <- igraph::sample_gnp(50, p[i,1])
  G2[[i]] <- igraph::sample_gnp(50, p[i,2])
}
graph.cor.test(G1, G2)
```

---

graph.entropy *Graph spectral entropy*

Description

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

Usage

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

Arguments

G	the undirected graph (igraph type) or its adjacency matrix. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative real values 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", "bcv", "ucv" and "SJ". "bcv" is an abbreviation of biased cross-validation, while "ucv" means unbiased cross-validation. "SJ" implements the methods of Sheather & Jones (1991) to select the bandwidth using pilot estimation of derivatives.
eigenvalues	optional parameter. It contains the eigenvalues of matrix G. Then, if the eigenvalues of matrix G have already been computed, this parameter can be used instead of G. If no value is passed, then the eigenvalues of G will be computed by 'graph.entropy'.

Value

A list with class "statGraph" containing the following components:

method	a string indicating the used method.
info	a string showing details about the method.
data.name	a string with the data's name(s).
entropy	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.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. _Journal of the Royal Statistical Society series B_, 53, 683-690. http://www.jstor.org/stable/2345597.

Examples

```
G <- igraph::sample_gnp(n=100, p=0.5)
entropy <- graph.entropy(G)
entropy
```

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

Description

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

Usage

```
graph.hclust(G, k, method = "complete", bandwidth = "Silverman")
```

Arguments

G	a list of undirected graphs (igraph type) or their adjacency matrices. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative 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 showing which criterion is used to choose the bandwidth during the spectral density estimation. Choose between the following criteria: "Silverman" (default), "Sturges", "bcv", "ucv" and "SJ". "bcv" is an abbreviation of biased cross-validation, while "ucv" means unbiased cross-validation. "SJ" implements the methods of Sheather & Jones (1991) to select the bandwidth using pilot estimation of derivatives.

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.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. _Journal of the Royal Statistical Society series B_, 53, 683-690. http://www.jstor.org/stable/2345597.

Examples

```
set.seed(1)
G <- list()
for (i in 1:5){
  G[[i]] <- igraph::sample_gnp(50, 0.5)
}
for (i in 6:10){
  G[[i]] <- igraph::sample_smallworld(1, 50, 8, 0.2)
}
for (i in 11:15){
  G[[i]] <- igraph::sample_pa(50, power = 1, directed = FALSE)
}
graph.hclust(G, 3)
```

---

graph.kmeans *K-means for Graphs*

Description

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

Usage

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

Arguments

x	a list containing the graphs or their adjacency matrices 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 list with class "statGraph" containing the following components:

method	a string indicating the used method.
info	a string showing details about the method.
data.name	a string with the data's name(s).
cluster	a vector of the same length of x 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

```
set.seed(42)
g <- list()
for(i in 1:5){
  g[[i]] <- igraph::sample_gnp(30, p=0.2)
}
for(i in 6:10){
  g[[i]] <- igraph::sample_gnp(30, p=0.5)
}
res <- graph.kmeans(g, k=2, nstart=2)
res
```

---

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(
  G,
  models = NULL,
  parameters = NULL,
  eps = 0.01,
  bandwidth = "Silverman",
  eigenvalues = NULL,
  ...
)
```

Arguments

G	the undirected graph (igraph type) or its adjacency matrix. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative 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 showing which criterion is used to choose the bandwidth during the spectral density estimation. Choose between the following criteria: "Silverman" (default), "Sturges", "bcv", "ucv" and "SJ". "bcv" is an abbreviation of biased cross-validation, while "ucv" means unbiased cross-validation. "SJ" implements the methods of Sheather & Jones (1991) to select the bandwidth using pilot estimation of derivatives.

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

`...` additional arguments to be used for `graph.param.estimator()` function.

Value

A list with class "statGraph" containing the following components:

`method`	a string indicating the used method.
`info`	a string showing details about the method.
`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 "param" 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.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. _Journal of the Royal Statistical Society series B_, 53, 683-690. http://www.jstor.org/stable/2345597.

Examples

```
## Example using an igraph object as input data
set.seed(1)
G <- igraph::sample_gnp(n=30, p=0.5)

# Using strings to indicate the graph models
result1 <- graph.model.selection(G, models=c("ER", "WS"), eps=0.5)
result1


## Using functions to describe the graph models
# Erdos-Renyi graph
model1 <- function(n, p){
  return(igraph::sample_gnp(n, p))
}
# Watts-Strogatz small-world graph
model2 <- function(n, pr, K=8){
  return(igraph::sample_smallworld(1, n, K, pr))
}
parameters <- list(seq(0.01, 0.99, 0.49), seq(0.01, 0.99, 0.49))
result2 <- graph.model.selection(G, 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(
  G,
  plot = TRUE,
  bandwidth = "Silverman",
  type = "n",
  main = "",
  ...
)
```

Arguments

`G`	a list of undirected graphs (igraph type) or their adjacency matrices. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative 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`	string showing which criterion is used to choose the bandwidth during the spectral density estimation. Choose between the following criteria: "Silverman" (default), "Sturges", "bcv", "ucv" and "SJ". "bcv" is an abbreviation of biased cross-validation, while "ucv" means unbiased cross-validation. "SJ" implements the methods of Sheather & Jones (1991) to select the bandwidth using pilot estimation of derivatives.
`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 list with class "statGraph" containing the following components:

`method`	a string indicating the used method.

info	a string showing details about the method.
data.name	a string with the data's name(s).
values	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.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. _Journal of the Royal Statistical Society series B_, 53, 683-690. http://www.jstor.org/stable/2345597.

Examples

```
set.seed(1)
G <- list()
for (i in 1:5){
  G[[i]] <- igraph::sample_gnp(50, 0.5)
}
for (i in 6:10){
  G[[i]] <- igraph::sample_smallworld(1, 50, 8, 0.2)
}
for (i in 11:15){
  G[[i]] <- igraph::sample_pa(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(
  G,
  model,
  parameters = NULL,
  eps = 0.01,
```

```
  bandwidth = "Silverman",
  eigenvalues = NULL,
  spectra = NULL,
  classic = TRUE,
  npoints = 2000,
  numCores = 1
)
```

Arguments

G	the undirected graph (igraph type) or its adjacency matrix. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have 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 Kullback-Leiber divergence or L1 norm. 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. The default 'parameter' is also used when `classic` = FALSE.
eps	precision of the grid (default is 0.01) when 'classic' is TRUE.
bandwidth	string showing which criterion is used to choose the bandwidth during the spectral density estimation. Choose between the following criteria: "Silverman" (default), "Sturges", "bcv", "ucv" and "SJ". "bcv" is an abbreviation of biased cross-validation, while "ucv" means unbiased cross-validation. "SJ" implements the methods of Sheather & Jones (1991) to select the bandwidth using pilot estimation of derivatives.

`eigenvalues` optional parameter. It contains the eigenvalues of matrix G. Then, it can be used when the eigenvalues of G were previously computed. If no value is passed, then the eigenvalues of G 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 parameter is estimated using the graph parameter estimator, where this option works better for large graphs with 1000 or more nodes. If TRUE (default) parameter is estimated using grid search.

`npoints` Number of points to discretize the interval of the largest and smallest eigenvalue of the graph's adjacency matrix eigenvalues. Only used when `classic` = FALSE.

`numCores` Number of cores to use for parallelization. Only used when `classic` = FALSE.

Value

A list with class "statGraph" containing the following components:

`method` a string indicating the used method.

`info` a string showing details about the method.

`data.name` a string with the data's name(s).

`param` 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.

`KLD` the Kullback-Leibler divergence between the observed graph and the graph model with the estimated parameter. Only returned if `classic` = TRUE.

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.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. _Journal of the Royal Statistical Society series B_, 53, 683-690. http://www.jstor.org/stable/2345597.

Examples

```
set.seed(1)
G <- igraph::sample_gnp(n=50, p=0.5)

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

## Not run:
# Using a function to describe the graph model
# Erdos-Renyi graph
set.seed(1)
model <- function(n, p){
  return(igraph::sample_gnp(n, p))
}
result2 <- graph.param.estimator(G, model,  seq(0.2, 0.8, 0.1))
result2


### 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
result3 <- graph.param.estimator(G, "WS", eps = 1e-1, npoints = 10,
                                 numCores = 1)
result3

## End(Not run)
```

`graph.parameter.estimator` *Degree-based graph parameter estimator*

Description

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

Usage

```
graph.parameter.estimator(
  G,
  model,
  lo = NULL,
  hi = NULL,
  eps = 0.001,
  from = NULL,
  to = NULL,
```

```
  npoints = 2000,
  numCores = 1
)
```

Arguments

G	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

A list with class "statGraph" containing the following components:

method	a string indicating the used method.
info	a string showing details about the method.
data.name	a string with the data's name(s).
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.
L1_dist	The L1 distance between the observed graph and the graph model with the estimated value.

Examples

```
set.seed(42)

### 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.parameter1 <- graph.parameter.estimator(G, "WS", lo=0.1, hi=0.5, eps=1e-1, npoints=10, numCores=1)
estimated.parameter1

## Not run:
### Example giving a function instead of a model

# 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.parameter2 <- graph.parameter.estimator(G, fun_WS, lo=0.0, hi=1.0, npoints=100, numCores=2)
estimated.parameter2

## End(Not run)
```

---

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(
  G,
  model,
  maxBoot = 500,
  spectra = NULL,
  eps = 0.01,
  classic = TRUE,
  bandwidth = "Silverman"
)
```

Arguments

G	a list of undirected graphs (igraph type) or their adjacency matrices. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative 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 parameter is estimated using the graph parameter estimator, where this option works better for large graphs with 1000 or more nodes. If TRUE (default) parameter is estimated using grid search.
bandwidth	string showing which criterion is used to choose the bandwidth during the spectral density estimation. Choose between the following criteria: "Silverman" (default), "Sturges", "bcv", "ucv" and "SJ". "bcv" is an abbreviation of biased cross-validation, while "ucv" means unbiased cross-validation. "SJ" implements the methods of Sheather & Jones (1991) to select the bandwidth using pilot estimation of derivatives.

Value

A list with class "htest" containing the following components:

statistic	the F statistic of the test.
p.value	the p-value of the test.
method	a string indicating the used method.
data.name	a string with the data's name(s).
estimates	a vector containing the estimated parameters for each graph.

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

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. _Journal of the Royal Statistical Society series B_, 53, 683-690. http://www.jstor.org/stable/2345597.

Examples

```
## Not run:
set.seed(42)
model <- "ER"
G <- list()

# Under H0
G[[1]] <- igraph::sample_gnp(50, 0.5)
G[[2]] <- igraph::sample_gnp(50, 0.5)
G[[3]] <- igraph::sample_gnp(50, 0.5)
result1 <- sp.anogva(G, model, maxBoot = 300)
result1

# Under H1
G[[1]] <- igraph::sample_gnp(50, 0.5)
G[[2]] <- igraph::sample_gnp(50, 0.55)
G[[3]] <- igraph::sample_gnp(50, 0.5)
result2 <- sp.anogva(G, model, maxBoot = 300)
result2

## End(Not run)
```

---

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

Description

`takahashi.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

```
takahashi.test(G1, G2, maxBoot = 1000, bandwidth = "Silverman")
```

Arguments

`G1`	a list of undirected graphs (igraph type) or their adjacency matrices. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative real values that correspond to the weights of the edges.
`G2`	a list of undirected graphs (igraph type) or their adjacency matrices. The adjacency matrix of an unweighted graph contains only 0s and 1s, while the weighted graph may have nonnegative real values that correspond to the weights of the edges.
`maxBoot`	integer indicating the number of bootstrap resamplings.

`bandwidth` string showing which criterion is used to choose the bandwidth during the spectral density estimation. Choose between the following criteria: "Silverman" (default), "Sturges", "bcv", "ucv" and "SJ". "bcv" is an abbreviation of biased cross-validation, while "ucv" means unbiased cross-validation. "SJ" implements the methods of Sheather & Jones (1991) to select the bandwidth using pilot estimation of derivatives.

Details

Given two lists of graphs, 'G1' and 'G2', 'takahashi.test' tests H0: "JS divergence between 'G1' and 'G2' is 0" against H1: "JS divergence between 'G1' and 'G2' is larger than 0".

Value

A list with class "htest" containing the following components:

`statistic`	the value of the Jensen-Shannon divergence (JSD) between 'G1' and 'G2'.
`p.value`	the p-value of the test.
`method`	a string indicating the used method.
`data.name`	a string with the data's name(s).

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.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. _Journal of the Royal Statistical Society series B_, 53, 683-690. http://www.jstor.org/stable/2345597.

Examples

```
set.seed(1)
G1 <- G2 <- list()
for (i in 1:20){
  G1[[i]] <- igraph::sample_gnp(n=50, p=0.5)
  G2[[i]] <- igraph::sample_gnp(n=50, p=0.51)
}
result <- takahashi.test(G1, G2, maxBoot=100)
result
```

`tang.test` *Tang hypothesis testing for random graphs.*

Description

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

Usage

```
tang.test(G1, G2, dim, sigma = NULL, maxBoot = 200)
```

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.
`maxBoot`	integer indicating the number of bootstrap resamples (default is 200).

Value

A list with class "htest" containing the following components:

`statistic`	the T-value of the test.
`p.value`	the p-value of the test.
`method`	a string indicating the used method.
`data.name`	a string with the data's name(s).

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

```
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 <- igraph::sample_dot_product(lpvs)
G2 <- igraph::sample_dot_product(lpvs)
```

```
D1 <- tang.test(G1, G2, 5)
D1

## test under H1
lpvs2 <- matrix(pnorm(200), 20, 10)
lpvs2 <- apply(lpvs2, 2, function(x){ return (abs(x)/sqrt(sum(x^2))) })
G2 <- suppressWarnings(igraph::sample_dot_product(lpvs2))
D2 <- tang.test(G1, G2, 5)
D2
```

Index