# Package 'NetworkDistance'

August 21, 2021

**Type** Package

**Title** Distance Measures for Networks

**Version** 0.3.4

**Description** Network is a prevalent form of data structure in many fields. As an object of analysis, many distance or metric measures have been proposed to define the concept of similarity between two networks. We provide a number of distance measures for networks. See Jurman et al (2011) <doi:10.3233/978-1-60750-692-8-227> for an overview on spectral class of inter-graph distance measures.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 3.0.0)

**Imports** Matrix, Rcpp, Rdpack, RSpectra, doParallel, foreach, graphon, parallel, stats, igraph, network, pracma, utils

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** graphics, knitr, rmarkdown

**RdMacros** Rdpack

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Kisung You [aut, cre] (<https://orcid.org/0000-0002-8584-459X>)

**Maintainer** Kisung You <kisungyou@outlook.com>

**Repository** CRAN

**Date/Publication** 2021-08-21 15:00:08 UTC

## R topics documented:

---

`graph20` *20 adjacency matrices from Erdős–Rényi models*

---

## Description

Simulated list of 20 adjacency matrices of 28 nodes. First 10 are from Erdős–Rényi model with $p = 0.9$, and the latter 10 are generated using $p = 0.5$. Each element in the list is of size $(28 \times 28)$, symmetric, having values in $0$ or $1$, and every diagonal element is set as $0$ in accordance with no self-loop assumption.

## Usage

```
data(graph20)
```

## Format

A `list` of 20 adjacency matrices of size $(28 \times 28)$.

## Details

Below is the code used to generate *graph20*:

```
require(stats)
graph20 = list()
for (i in 1:10){ # type-1 adjacency matrices
  rbin   = rbinom(784,1,0.9)
  mat    = matrix(rbin, nrow=28)
  matout = mat*t(mat)
  diag(matout) = 0
  graph20[[i]]=matout
}
for (i in 11:20){ # type-2 adjacency matrices
  rbin   = rbinom(784,1,0.5)
  mat    = matrix(rbin, nrow=28)
  matout = mat*t(mat)
```

```
  diag(matout) = 0
  graph20[[i]]=matout
}
```

---

`nd.centrality` *Centrality Distance*

---

## Description

Centrality is a core concept in studying the topological structure of complex networks, which can be either defined for each node or edge. `nd.centrality` offers 3 distance measures on node-defined centralities. See this Wikipedia page for more on network/graph centrality.

## Usage

```
nd.centrality(
  A,
  out.dist = TRUE,
  mode = c("Degree", "Close", "Between"),
  directed = FALSE
)
```

## Arguments

| | |
|---|---|
| `A` | a list of length $N$ containing $(M \times M)$ adjacency matrices. |
| `out.dist` | a logical; `TRUE` for computed distance matrix as a `dist` object. |
| `mode` | type of node centrality definitions to be used. |
| `directed` | a logical; `FALSE` as symmetric, undirected graph. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or `dist` object containing pairwise distance measures.

**features** an $(N \times M)$ matrix where rows are node centralities for each graph.

## References

Roy M, Schmid S, Trédan G (2014). "Modeling and Measuring Graph Similarity: The Case for Centrality Distance." In *FOMC 2014, 10th ACM International Workshop on Foundations of Mobile Computing*, 53.

## Examples

```
## load example data
data(graph20)

## use 3 types of centrality measures
out1 <- nd.centrality(graph20, out.dist=FALSE,mode="Degree")
out2 <- nd.centrality(graph20, out.dist=FALSE,mode="Close")
out3 <- nd.centrality(graph20, out.dist=FALSE,mode="Between")

## visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(1,3), pty="s")
image(out1$D[,20:1], main="Degree", col=gray(0:32/32), axes=FALSE)
image(out2$D[,20:1], main="Close", col=gray(0:32/32), axes=FALSE)
image(out3$D[,20:1], main="Between", col=gray(0:32/32), axes=FALSE)
par(opar)
```

---

# nd.csd *$L\_2$ Distance of Continuous Spectral Densities*

---

## Description

The method employs spectral density of eigenvalues from Laplacian in that for each, we have corresponding spectral density $\rho(w)$ as a sum of narrow Lorentz distributions with `bandwidth` parameter. Since it involves integration of a function over the non-compact domain, it may blow up to infinity and the code automatically aborts the process.

## Usage

```
nd.csd(A, out.dist = TRUE, bandwidth = 1)
```

## Arguments

| | |
|---|---|
| `A` | a list of length $N$ containing $(M \times M)$ adjacency matrices. |
| `out.dist` | a logical; `TRUE` for computed distance matrix as a `dist` object. |
| `bandwidth` | common bandwidth of positive real number. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or `dist` object containing pairwise distance measures.

**spectra** an $(N \times M - 1)$ matrix where each row is top-$M - 1$ vibrational spectra.

## References

Ipsen M, Mikhailov AS (2002). "Evolutionary reconstruction of networks." *Physical Review E*, **66**(4). ISSN 1063-651X, 1095-3787.

## Examples

```
## load example data
data(graph20)

## compute distance matrix
output = nd.csd(graph20, out.dist=FALSE, bandwidth=1.0)

## visualize
opar = par(no.readonly=TRUE)
par(pty="s")
image(output$D[,20:1], main="two group case", axes=FALSE, col=gray(0:32/32))
par(opar)
```

---

`nd.dsd` *Discrete Spectral Distance*

---

## Description

Discrete Spectral Distance (DSD) is defined as the Euclidean distance between the spectra of various matrices, such as adjacency matrix $A$("Adj"), (unnormalized) Laplacian matrix $L = D - A$("Lap"), signless Laplacian matrix $|L| = D + A$("SLap"), or normalized Laplacian matrix $\tilde{L} = D^{-1/2}LD^{-1/2}$.

## Usage

```
nd.dsd(A, out.dist = TRUE, type = c("Lap", "SLap", "NLap", "Adj"))
```

## Arguments

| | |
|---|---|
| `A` | a list of length $N$ containing $(M \times M)$ adjacency matrices. |
| `out.dist` | a logical; `TRUE` for computed distance matrix as a `dist` object. |
| `type` | type of target structure. One of `"Lap"`, `"SLap"`, `"NLap"`, `"Adj"` as defined above. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or `dist` object containing pairwise distance measures.

**spectra** an $(N \times M - 1)$ matrix where each row is top-$M - 1$ vibrational spectra.

## References

Wilson RC, Zhu P (2008). "A study of graph spectra for comparing graphs and trees." *Pattern Recognition*, **41**(9), 2833–2841. ISSN 00313203.

## Examples

```
## load example data and extract only a few
data(graph20)
gr.small = graph20[c(1:5,11:15)]

## compute distance matrix
output <- nd.dsd(gr.small, out.dist=FALSE)

## visualize
opar <- par(no.readonly=TRUE)
par(pty="s")
image(output$D[,10:1], main="two group case", axes=FALSE, col=gray(0:32/32))
par(opar)
```

---

`nd.edd` *Edge Difference Distance*

---

## Description

It is of the most simplest form that Edge Difference Distance (EDD) takestwo adjacency matrices and takes Frobenius norm of their differnces.

## Usage

```
nd.edd(A, out.dist = TRUE)
```

## Arguments

| | |
|---|---|
| `A` | a list of length $N$ containing $(M \times M)$ adjacency matrices. |
| `out.dist` | a logical; `TRUE` for computed distance matrix as a `dist` object. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or `dist` object containing pairwise distance measures.

## References

Hammond DK, Gur Y, Johnson CR (2013). "Graph Diffusion Distance: A Difference Measure for Weighted Graphs Based on the Graph Laplacian Exponential Kernel." In *Proceedings of the IEEE global conference on information and signal processing (GlobalSIP'13)*, 419–422.

## Examples

```
## load example data
data(graph20)

## compute distance matrix
output = nd.edd(graph20, out.dist=FALSE)

## visualize
opar <- par(no.readonly=TRUE)
par(pty="s")
image(output$D[,20:1], main="two group case", axes=FALSE, col=gray(0:32/32))
par(opar)
```

---

`nd.extremal` *Extremal distance with top-$k$ eigenvalues*

---

## Description

Extremal distance (`nd.extremal`) is a type of spectral distance measures on two graphs' graph Laplacian,

$$L := D - A$$

where $A$ is an adjacency matrix and $D_{ii} = \sum_j A_{ij}$. It takes top-$k$ eigenvalues from graph Laplacian matrices and take normalized sum of squared differences as metric. Note that it is *1. non-negative*, *2. separated*, *3. symmetric*, and satisfies *4. triangle inequality* in that it is indeed a metric.

## Usage

```
nd.extremal(A, out.dist = TRUE, k = ceiling(nrow(A)/5))
```

## Arguments

| | |
|---|---|
| `A` | a list of length `N` containing adjacency matrices. |
| `out.dist` | a logical; `TRUE` for computed distance matrix as a `dist` object. |
| `k` | the number of largest eigenvalues to be used. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or `dist` object containing pairwise distance measures.

**spectra** an $(N \times k)$ matrix where each row is top-$k$ Laplacian eigenvalues.

## References

Jakobson D, Rivin I (2002). "Extremal metrics on graphs I." *Forum Mathematicum*, **14**(1). ISSN 0933-7741, 1435-5337.

## Examples

```
## load data
data(graph20)

## compute distance matrix
output = nd.extremal(graph20, out.dist=FALSE, k=2)

## visualize
opar = par(no.readonly=TRUE)
par(pty="s")
image(output$D[,20:1], main="two group case", col=gray(0:32/32), axes=FALSE)
par(opar)
```

---

nd.gdd *Graph Diffusion Distance*

---

## Description

Graph Diffusion Distance (nd.gdd) quantifies the difference between two weighted graphs of same size. It takes an idea from heat diffusion process on graphs via graph Laplacian exponential kernel matrices. For a given adjacency matrix $A$, the graph Laplacian is defined as

$$L := D - A$$

where $D_{ii} = \sum_j A_{ij}$. For two adjacency matrices $A_1$ and $A_2$, GDD is defined as

$$d_{gdd}(A_1, A_2) = max_t \sqrt{\| \exp(-tL_1) - \exp(-tL_2) \|_F^2}$$

where $\exp(\cdot)$ is matrix exponential, $\| \cdot \|_F$ a Frobenius norm, and $L_1$ and $L_2$ Laplacian matrices corresponding to $A_1$ and $A_2$, respectively.

## Usage

```
nd.gdd(A, out.dist = TRUE, vect = seq(from = 0.1, to = 1, length.out = 10))
```

## Arguments

| | |
|---|---|
| A | a list of length $N$ containing adjacency matrices. |
| out.dist | a logical; TRUE for computed distance matrix as a dist object. |
| vect | a vector of parameters $t$ whose values will be used. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or dist object containing pairwise distance measures.

**maxt** an $(N \times N)$ matrix whose entries are maximizer of the cost function.

## References

Hammond DK, Gur Y, Johnson CR (2013). "Graph Diffusion Distance: A Difference Measure for Weighted Graphs Based on the Graph Laplacian Exponential Kernel." In *Proceedings of the IEEE global conference on information and signal processing (GlobalSIP'13)*, 419–422.

## Examples

```
## load data and extract a subset
data(graph20)
gr.small = graph20[c(1:5,11:15)]

## compute distance matrix
output = nd.gdd(gr.small, out.dist=FALSE)

## visualize
opar = par(no.readonly=TRUE)
par(pty="s")
image(output$D[,10:1], main="two group case", col=gray((0:32)/32), axes=FALSE)
par(opar)
```

---

`nd.graphon` *Graphon Estimates Distance*

---

## Description

Graphon is a symmetric measurable function

$$W : [0, 1]^2 \to [0, 1]$$

that is considered to be a generating model for an observed network. `nd.graphon` computes distances between networks based on the estimated graphons of each network. Estimation methods are taken from **graphon** package. For more details, see the function links below.

## Usage

```
nd.graphon(
  A,
  out.dist = TRUE,
  method = c("completion", "LG", "nbdsmooth", "SBA", "USVT"),
  ...
)
```

## Arguments

| | |
|---|---|
| `A` | a list of length $N$ containing $(M \times M)$ adjacency matrices. |
| `out.dist` | a logical; `TRUE` for computed distance matrix as a `dist` object. |
| `method` | type of graphon estimation methods to be used. |

... extra parameters to be passed onto graphon estimation functions. See also est.completion, est.LG, est.nbdsmooth, est.SBA, and est.USVT for details.

## Value

a named list containing

**D** an $(N \times N)$ matrix or dist object containing pairwise distance measures.

## References

Mukherjee SS, Sarkar P, Lin L (2017). "On clustering network-valued data." In Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds.), *Advances in neural information processing systems 30*, 7071–7081. Curran Associates, Inc.

## Examples

```
## load example data
data(graph20)

## compute USVT-based distance
output <- nd.graphon(graph20, out.dist=FALSE, method="usvt")

## visualize
opar = par(no.readonly=TRUE)
par(pty="s")
image(output$D[,20:1], main="USVT", col=gray(0:32/32), axes=FALSE)
par(opar)
```

---

# nd.hamming *Hamming Distance*

---

## Description

Hamming Distance is the count of discrepancy between two binary networks for each edge. Therefore, if used with non-binary networks, it might return a warning message and distorted results. It was originally designed to compare two strings of equal length, see Wikipedia page for more detailed introduction.

## Usage

```
nd.hamming(A, out.dist = TRUE)
```

## Arguments

| | |
|---|---|
| A | a list of length $N$ containing adjacency matrices. |
| out.dist | a logical; TRUE for computed distance matrix as a dist object. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or `dist` object containing pairwise distance measures.

## References

Hamming RW (1950). "Error Detecting and Error Correcting Codes." *Bell System Technical Journal*, **29**(2), 147–160. ISSN 00058580.

## Examples

```
## load example data and extract only a few
data(graph20)
gr.small = graph20[c(1:5,11:15)]

## compute distance matrix
output = nd.hamming(gr.small, out.dist=FALSE)

## visualize
opar = par(no.readonly=TRUE)
par(pty="s")
image(output$D[,10:1], main="two group case", axes=FALSE, col=gray(0:32/32))
par(opar)
```

---

`nd.him` *HIM Distance*

---

## Description

Hamming-Ipsen-Mikhailov (HIM) combines the local Hamming edit distance and the global Ipsen-Mikhailov distance to merge information at each scale. For Ipsen-Mikhailove distance, it is provided as `nd.csd` in our package for consistency. Given a parameter $\xi$ (`xi`), it is defined as

$$HIM_{\xi}(A,B) = \sqrt{H^2(A,B) + \xi \cdot IM^2(A,B)}/\sqrt{1+\xi}$$

where $H$ and $IM$ stand for Hamming and I-M distance, respectively.

## Usage

```
nd.him(A, out.dist = TRUE, xi = 1, ntest = 10)
```

## Arguments

| | |
|---|---|
| `A` | a list of length $N$ containing $(M \times M)$ adjacency matrices. |
| `out.dist` | a logical; `TRUE` for computed distance matrix as a `dist` object. |
| `xi` | a parameter to control balance between two distances. |
| `ntest` | the number of searching over `nd.csd` parameter. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or `dist` object containing pairwise distance measures.

## References

Jurman G, Visintainer R, Filosi M, Riccadonna S, Furlanello C (2015). "The HIM glocal metric and kernel for network comparison and classification." In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 1–10. ISBN 978-1-4673-8272-4.

## See Also

`nd.hamming`, `nd.csd`

## Examples

```
## load example data
data(graph20)

## compute distance matrix
output = nd.him(graph20, out.dist=FALSE)

## visualize
opar = par(no.readonly=TRUE)
par(pty="s")
image(output$D[,20:1], main="two group case", axes=FALSE, col=gray(0:32/32))
par(opar)
```

---

`nd.moments` *Log Moments Distance*

---

## Description

For a graph with an adjacency matrix $A$, *graph moment* is defined as

$$\rho_m(A) = tr(A/n)^m$$

where $n$ is the number of vertices and $m$ is an order of the moment. `nd.moments` computes pairwise distances based on log of graph moments from $m = 1$ to $m = k$.

## Usage

```
nd.moments(
  A,
  k = 3,
 metric = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  out.dist = TRUE
)
```

## Arguments

| | |
|---|---|
| A | a list of length $N$ containing $(M \times M)$ adjacency matrices. |
| k | the integer order of moments. If $k$ is too large, it may incur numerical overflow. |
| metric | type of distance measures for log-moment features. See dist for more details. |
| out.dist | a logical; TRUE for computed distance matrix as a dist object. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or dist object containing pairwise distance measures.

## References

Mukherjee SS, Sarkar P, Lin L (2017). "On clustering network-valued data." In Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds.), *Advances in neural information processing systems 30*, 7071–7081. Curran Associates, Inc.

## Examples

```
## load example data
data(graph20)

## compute distance based on different k's.
out3 <- nd.moments(graph20, k=3, out.dist=FALSE)
out5 <- nd.moments(graph20, k=5, out.dist=FALSE)
out7 <- nd.moments(graph20, k=7, out.dist=FALSE)
out9 <- nd.moments(graph20, k=9, out.dist=FALSE)

## visualize
opar = par(no.readonly=TRUE)
par(mfrow=c(2,2), pty="s")
image(out3$D[,20:1], col=gray(0:32/32), axes=FALSE, main="k=3")
image(out5$D[,20:1], col=gray(0:32/32), axes=FALSE, main="k=5")
image(out7$D[,20:1], col=gray(0:32/32), axes=FALSE, main="k=7")
image(out9$D[,20:1], col=gray(0:32/32), axes=FALSE, main="k=9")
par(opar)
```

nd.nfd *Network Flow Distance*

## Description

Network Flow Distance

## Usage

```
nd.nfd(
  A,
  order = 0,
  out.dist = TRUE,
  vect = seq(from = 0, to = 10, length.out = 1000)
)
```

## Arguments

| | |
|---|---|
| A | a list of length $N$ containing adjacency matrices. |
| order | the order of Laplacian; currently only 0 and 1 are supported. |
| out.dist | a logical; TRUE for computed distance matrix as a dist object. |
| vect | a vector of parameters $t$ whose values will be used. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or dist object containing pairwise distance measures.

## Examples

```
## Not run:
## load example data
data(graph20)

# compute two diffusion-based distances and visualize
out1 = nd.gdd(graph20, out.dist=FALSE)
out2 = nd.nfd(graph20, out.dist=FALSE)

# visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2), pty="s")
image(out1$D[,20:1],col=gray((0:32)/32), main="nd.gdd",axes=FALSE)
image(out2$D[,20:1],col=gray((0:32)/32), main="nd.nfd",axes=FALSE)
par(opar)

## End(Not run)
```

nd.wsd *Distance with Weighted Spectral Distribution*

## Description

Normalized Laplacian matrix contains topological information of a corresponding network via its spectrum. nd.wsd adopts weighted spectral distribution of eigenvalues and brings about a metric via binning strategy.

## Usage

```
nd.wsd(A, out.dist = TRUE, K = 50, wN = 4)
```

## Arguments

| | |
|---|---|
| A | a list of length $N$ containing $(M \times M)$ adjacency matrices. |
| out.dist | a logical; TRUE for computed distance matrix as a dist object. |
| K | the number of bins for the spectrum interval $[0, 2]$. |
| wN | a decaying exponent; default is $4$ set by authors. |

## Value

a named list containing

**D** an $(N \times N)$ matrix or dist object containing pairwise distance measures.

**spectra** an $(N \times M)$ matrix of rows being eigenvalues for each graph.

## References

Fay D, Haddadi H, Thomason A, Moore AW, Mortier R, Jamakovic A, Uhlig S, Rio M (2010). "Weighted Spectral Distribution for Internet Topology Analysis: Theory and Applications." *IEEE/ACM Transactions on Networking*, **18**(1), 164–176. ISSN 1063-6692, 1558-2566.

## Examples

```
## load example data and extract a few
data(graph20)
gr.small = graph20[c(1:5,11:15)]

## compute distance matrix
output = nd.wsd(gr.small, out.dist=FALSE, K=10)

## visualize
opar = par(no.readonly=TRUE)
par(pty="s")
image(output$D[,10:1], main="two group case", axes=FALSE, col=gray(0:32/32))
par(opar)
```

NetworkDistance *Distance Measures for Networks*

## Description

Network has gathered much attention from many disciplines, as many of real data can be well represented in the relational form. The concept of distance - or, metric - between two networks is the starting point for inference on population of networks. **NetworkDistance** package provides a not-so-comprehensive collection of distance measures for measuring dissimilarity between two network objects. Data should be supplied as *adjacency* matrices, where we support three formats of data representation; `matrix` object in **R** base, `network` class from **network** package, and `igraph` class from **igraph** package.

# Index