## Package ‘oddnet’

December 22, 2022

**Type**  Package  
**Title**  Anomaly Detection in Temporal Networks  
**Version**  0.1.0  
**Maintainer**  Sevvandi Kandanaarachchi <sevvandik@gmail.com>  
**Description**  Anomaly detection in dynamic, temporal networks. The package 'oddnet' uses a feature-based method to identify anomalies. First, it computes many features for each network. Then it models the features using time series methods. Using time series residuals it detects anomalies. This way, the temporal dependencies are accounted for when identifying anomalies (Kandanaarachchi, Hyndman 2022) <arXiv:2210.07407>.  
**License**  GPL (>= 3)  
**Encoding**  UTF-8  
**RoxygenNote**  7.2.1  
**Imports**  dplyr, fable, fabletools, igraph, lookout, pcaPP, rlang, tibble, tidyr, tsibble, utils  
**Suggests**  DDoutlier, feasts, knitr, rmarkdown, rTensor, urca  
**VignetteBuilder**  knitr  
**URL**  https://sevvandi.github.io/oddnet/  
**NeedsCompilation**  no  
**Author**  Sevvandi Kandanaarachchi [aut, cre]  
(https://orcid.org/0000-0002-0337-0395),  
Rob Hyndman [aut] (https://orcid.org/0000-0002-2140-5352)  
**Repository**  CRAN  
**Date/Publication**  2022-12-22 20:10:01 UTC

### R topics documented:

- anomalous_networks .............................................. 2  
- compute_features .................................................. 3  
- lad ................................................................. 4  
- tensorsplat ........................................................ 5
anomalous_networks


Description

This function identifies anomalous networks from a series of temporal networks. It uses graph theoretic features to transform networks to a feature space. This function has parameters for feature computation, scaling, robust PCA and anomaly detection procedures.

Usage

anomalous_networks(
  networks,  # The input series of temporal networks given in a list with each network denoted by its adjacency matrix.
  alpha = 0.05,  # An anomaly detection parameter. The level of significance for the anomaly detection algorithm lookout. Default is 0.05.
  dd = 2,  # A robust PCA parameter. The number of reduced dimensions in robust PCA. Default is 2.
  trim = 0.005,  # A scaling parameter. The percentage used to compute trimmed mean and trimmed standard deviation. Default is 0.5 percent.
  na_action = NULL,  # The action for NA valued features.
  vert_attr = FALSE,  # A feature computation parameter. If TRUE the network nodes/vertices have attributes.
  attr_name = NULL,  # A feature computation parameter. The name of the network vertex attribute. Only a single attribute can be specified.
  attr_mat = NULL,  # A feature computation parameter. If network nodes/vertices have attributes, the list of attribute matrices for each network can be given using this feature.
  fast = FALSE  # If set to TRUE will avoid computing time consuming features.
)

Arguments

networks
alpha
dd
trim
na_action
vert_attr
attr_name
attr_mat
fast
**compute_features**

Value

Object imported from lookout.

See Also

[lookout::lookout()]

Examples

```r
# We generate a series of networks and add an anomaly at 50th network.
set.seed(1)
networks <- list()
p.or.m.seq <- rep(0.1, 50)
p.or.m.seq[20] <- 0.3  # anomalous network at 20
for(i in 1:50){
  gr <- igraph::erdos.renyi.game(50, p.or.m = p.or.m.seq[i])
networks[[i]] <- igraph::as_adjacency_matrix(gr)
}
anomalous_networks(networks, fast = TRUE)
```

---

**compute_features**

*Computes features for each network.*

Description

This function computes features for each network using graph theoretic constructs.

Usage

```r
compute_features(gr, attributes = FALSE, attr_name = NULL, fast = FALSE)
```

Arguments

- `gr`:
  The network or graph as an igraph object.
- `attributes`:
  If the network nodes/vertices have attributes, then attributes = TRUE.
- `attr_name`:
  The name of the node/vertex attribute. Only a single attribute can be specified.
- `fast`:
  If set to TRUE will avoid computing time consuming features.

Value

A network features object containing 20 graph-theoretic features.
Examples

```r
set.seed(1)
gr <- igraph::erdos.renyi.game(100, 0.05)
compute_features(gr)
```

lad

*Laplacian Eigen Value method by Shenyang Huang, Yasmeen Hitti, Guillaume Rabusseau and Reihaneh Rabbany from their KDD’20 paper Laplacian Change Point Detection for Dynamic Graphs*

Description

Laplacian Eigen Value method by Shenyang Huang, Yasmeen Hitti, Guillaume Rabusseau and Reihaneh Rabbany from their KDD’20 paper Laplacian Change Point Detection for Dynamic Graphs

Usage

```r
lad(matlist, k = NULL, short_win, long_win, alpha = 0.05, from_file = NULL)
```

Arguments

- `matlist` The matrix list, where each matrix is an adjacency matrix of the graph.
- `k` The number of eigen values to consider
- `short_win` The length of the shorter windows
- `long_win` The length of the longer windows
- `alpha` The threshold to declare anomalies
- `from_file` This is an additional parameter only if a file needs to be read

Value

An object of class lad. LAD is a window based method. It considers short and a long windows. The lad object has anomalous scores when taking into account short and long windows along with the identified anomalies for both short and long windows.

References

Examples

# We generate a series of networks and add an anomaly at 50th network.
set.seed(1)
networks <- list()
p.or.m.seq <- rep(0.05, 50)
p.or.m.seq[20] <- 0.2 # anomalous network at 20
for(i in 1:50){
  gr <- igraph::erdos.renyi.game(100, p.or.m = p.or.m.seq[i])
  networks[[i]] <- igraph::as_adjacency_matrix(gr)
}
ladobj <- lad(networks, k = 6, short_win = 2, long_win = 4)
ladobj

tensorsplat  
\textit{Implements Danai Koutra's TensorSplat algorithm}

Description

Implements Danai Koutra’s TensorSplat algorithm

Usage

tensorsplat(matlist, k = 2, alpha = 0.05)

Arguments

- \texttt{matlist} The list of matrices where each matrix denotes the adjacency matrix of the network.
- \texttt{k} The number of components in PARFAC tensor decomposition.
- \texttt{alpha} The threshold to declare anomalies

Value

Anomalous observations

References

Examples

# We generate a series of networks and add an anomaly at 50th network.
set.seed(1)
networks <- list()
p.or.m.seq <- rep(0.05, 50)
p.or.m.seq[20] <- 0.2  # anomalous network at 20
for(i in 1:100){
  gr <- igraph::erdos.renyi.game(100, p.or.m = p.or.m.seq[i])
  networks[[i]] <- igraph::as_adjacency_matrix(gr)
}
tensobj <- tensorsplat(networks, k = 2)
tensobj  # anomalous networks
Index

anomalous_networks, 2
compute_features, 3
lad, 4
tensorsplat, 5