

# Package ‘oddnet’

July 22, 2026

**Type** Package

**Title** Anomaly Detection in Temporal Networks

**Version** 0.1.2

**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, Sanderson, Hyndman 2024) <doi:10.1109/ISCM163661.2024.10851659>.

**License** GPL (>= 3)

**Encoding** UTF-8

**Imports** dplyr, fable, fabletools, feasts, igraph, lookout, pcaPP, rlang, tibble, tidyr, tsibble, utils

**Suggests** knitr, rmarkdown, urca

**VignetteBuilder** knitr

**URL** <https://sevvandi.github.io/oddnet/>

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Sevvandi Kandanaarachchi [aut, cre] (ORCID: <<https://orcid.org/0000-0002-0337-0395>>),  
Rob Hyndman [aut] (ORCID: <<https://orcid.org/0000-0002-2140-5352>>)

**Repository** CRAN

**Date/Publication** 2026-07-22 06:00:02 UTC

## Contents

|                              |          |
|------------------------------|----------|
| anomalous_networks . . . . . | 2        |
| compute_features . . . . .   | 3        |
| lad . . . . .                | 4        |
| <b>Index</b>                 | <b>6</b> |

---

|                    |                                                                          |
|--------------------|--------------------------------------------------------------------------|
| anomalous_networks | <i>Identifies anomalous networks from a series of temporal networks.</i> |
|--------------------|--------------------------------------------------------------------------|

---

### 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. ADD MORE DESCRIPTION.

### Usage

```
anomalous_networks(
  networks,
  alpha = 0.05,
  dd = 2,
  trim = 0.005,
  na_action = NULL,
  vert_attr = FALSE,
  attr_name = NULL,
  attr_mat = NULL,
  fast = FALSE,
  feature_subset = NULL
)
```

### Arguments

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

**Value**

Object imported from lookout.

**See Also**

[lookout::lookout()]

**Examples**

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

---

|                  |                                            |
|------------------|--------------------------------------------|
| compute_features | <i>Computes features for each network.</i> |
|------------------|--------------------------------------------|

---

**Description**

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

**Usage**

```
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

```
set.seed(1)
gr <- igraph::sample_gnp(100, 0.05)
compute_features(gr)
```

---

|     |                                                                                                                                                                                            |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lad | <i>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</i> |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

---

## 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

```
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

Huang, S., Hitti, Y., Rabusseau, G., & Rabbany, R. (2020). Laplacian Change Point Detection for Dynamic Graphs. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 349–358. <https://doi.org/10.1145/3394486.3403077>

**Examples**

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

# Index

anomalous\_networks, [2](#)

compute\_features, [3](#)

lad, [4](#)