Package 'netseer'

February 27, 2026

Type Package

Title Graph Prediction from a Graph Time Series

Version 0.1.3

Maintainer Sevvandi Kandanaarachchi <sevvandik@gmail.com>

Description Predicting the structure of a graph including new nodes and edges using a time series of graphs. Flux balance analysis, a linear and integer programming technique used in biochemistry is used with time series prediction methods to predict the graph structure at a future time point Kandanaarachchi (2025) <doi:10.48550/arXiv.2507.05806>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports dplyr,
fable,
fabletools,
forecast,
future,
igraph,
lpSolve,
Matrix,
feasts,
rlang,
stats,
tibble,
tsibble

Suggests knitr,
nnet,
rmarkdown,
urca

URL https://sevvandi.github.io/netseer/

VignetteBuilder knitr

Depends R (>= 3.5)

Contents

`generate_graph_exp` *Generates a bigger graph using exponential growth.*

Description

Generates a bigger graph using parameters for node and edge growth. If a sequence of graphs are created, the number of nodes in this sequence would exponentially increase.

Usage

```
generate_graph_exp(
  gr = NULL,
  del_edge = 0.1,
  new_nodes = 0.1,
  edge_increase = 0.1
)
```

Arguments

`gr`	The input graph to generate the next graph. If set to `NULL` a graph using `igraph::sample_pa` is used as the input graph.
`del_edge`	The proportion of edges deleted from the input graph. Default set to `0.1`.
`new_nodes`	The proportion of nodes added to the input graph. Default set to `0.1`.
`edge_increase`	The proportion of edges added to the input graph. Default set to `0.1`.

Value

A graph.

Examples

```
set.seed(1)
gr <- generate_graph_exp()
gr
```

generate_graph_linear *Generates a bigger graph by linear growth.*

Description

Generates a bigger graph using parameters for node and edge growth. If a sequence of graphs are created, the number of nodes would linearly increase.

Usage

```
generate_graph_linear(
  gr = NULL,
  del_edge = 1,
  new_nodes = 1,
  edge_increase = 1,
  edges_per_new_node = 3
)
```

Arguments

gr	The input graph to generate the next graph. If set to NULL a graph using igraph::sample_pa is used as the input graph.
del_edge	The number of edges deleted from the input graph. Default set to 1.
new_nodes	The number of nodes added to the input graph. Default set to 1.
edge_increase	The number of edges added to the input graph. Default set to 1.
edges_per_new_node	The number of edges added to the new nodes. Default set to 3.

Value

A graph.

Examples

```
set.seed(1)
gr <- generate_graph_linear()
gr
```

generate_graph_list *Generates a bigger graph using either linear or exponential growth.*

Description

Wrapper around 'generate_graph_linear' and 'generate_graph_exp'.

Usage

```
generate_graph_list(
  num_graphs = 15,
  del_edge = 0.1,
  new_nodes = 0.1,
  edge_increase = 0.1,
  mode = "exp"
)
```

Arguments

num_graphs	The number of graphs to be generated. Default set to 15.
del_edge	The proportion of edges deleted from the input graph. Default set to 0.1.
new_nodes	The proportion of nodes added to the input graph. Default set to 0.1.
edge_increase	The proportion of edges added to the input graph. Default set to 0.1.
mode	The method the graphs grow in. Either "linear" for linear creation, or "exp". Default set to "exp".
gr	The input graph to generate the next graph. If set to NULL a graph using igraph::sample_pa is used as the input graph.

Value

A graph list.

Examples

```
set.seed(1)
gr <- generate_graph_list(12, 0.1, 0.1, 0.1, "exp")
gr
```

---

measure_error *Gives an error measurement for predicted graphs*

Description

This function compares the predicted graph with the actual and comptues the vertex and edge error as a proportion

Usage

```
measure_error(actual, predicted)
```

Arguments

actual	The ground truth or actual graph.
predicted	The predicted graph.

Value

The vertex error and edge error as a proportion.

Examples

```
data(syngraphs)
# Taking the 20th graph as the actual and the 19th graph as predicted.
measure_error(syngraphs[[20]], syngraphs[[19]])
```

`predict_graph` *Predicts a graph from a time series of graphs.*

Description

This function predicts the graph at a future time step using a time series of graphs.

Usage

```
predict_graph(
  graphlist,
  formulation = 2,
  conf_level1 = NULL,
  conf_level2 = 90,
  dense_opt = 2,
  weights_opt = 8,
  weights_param = 0.001,
  h = 1
)
```

Arguments

`graphlist`	A list of graphs in igraph format.
`formulation`	Formulation 2 includes an additional condition constraining total edges by the predicted value. Formulation 1 does not have that constraint. Formulation 2 gives more realistic graphs due to that constraint. Default is set to `2`.
`conf_level1`	A value between 50 and 100 denoting the confidence interval for the number of predicted nodes in the graph. If set to `NULL` the predicted graph has the mean number of predicted nodes. If set to `80` for example, there would be 3 predicted graphs. One with mean number of predicted nodes, and the other two with the number of nodes corresponding to lower and upper confidence bounds.
`conf_level2`	The upper confidence bound for the degree distribution. Default set to `90`.
`dense_opt`	If set to `2` the dense option in R package `lpSolve` will be used.
`weights_opt`	Weights option ranging from 1 to 6 used for different edge weight schemes. Weights option 1 uses uniform weights for all edges. Option 2 uses binary weights. If the edge existed in a past graph, then weight is set to 1. Else set to 0. All possible new edges are assigned weight 1. Option 3 is a more selective version. Option 4 uses proportional weights according to the history. Option 5 uses proportional weights, but as the network is more in the past, it gives less weight. Option 5 uses linearly decaying proportional weights. Option 6 uses harmonically decaying weights. That is the network at `T` is given weight 1, `T-1` is given weight 1/2 and so on. Option 7 uses 1 for edges that are present in the last graph. Option 8 is a slightly different to Option 7. It uses 1 for edges in the last seen graph and the `weights_param` for new edges. Default is set to `8`.

weights_param	The weight given for possible edges from new vertices. Default set to 0.001.
h	The prediction time step. Default is h = 1.

Value

A list of predicted graphs. If conf_level1 is not NULL, then 3 graphs are returned one with the mean number of predicted nodes and the other 2 with the number of nodes equal to the lower and upper bound values of prediction. If If conf_level1 is NULL, only the mean predicted graph is returned.

Examples

```
set.seed(2024)
edge_increase_val <- new_nodes_val <- del_edge_val <- 0.1
graphlist <- list()
graphlist[[1]] <- gr <-  igraph::sample_pa(5, directed = FALSE)
for(i in 2:15){
  gr <-  generate_graph_exp(gr,
                        del_edge = del_edge_val,
                        new_nodes = new_nodes_val,
                        edge_increase = edge_increase_val )
  graphlist[[i]] <- gr
}
grpred <- predict_graph(graphlist[1:15], conf_level2 = 90, weights_opt = 6)
grpred
```

---

read_graph_list *Reads graphs from a folder*

Description

This function reads graphs from a folder to a list

Usage

```
read_graph_list(path_to_graphs, format)
```

Arguments

path_to_graphs	The folder where graphs are contained.
format	Formats supported by igraph::read_graph.

Value

A list of graphs in igraph format.

Examples

```
path_to_graphs <- system.file("extdata", package = "netseer")
grlist <- read_graph_list(path_to_graphs = path_to_graphs, format = "gml")
grlist
```

syngraphs *A dataset containing synthetic graphs*

Description

This dataset contains a list of synthetic igraph objects

Usage

```
syngraphs
```

Format

A list of 20 igraph graphs

Examples

```
data(syngraphs)
syngraphs[[1]]
```

Index