Package: GNAR (via r-universe)

September 14, 2024

Type Package

Title Methods for Fitting Network Time Series Models

Version 1.1.3

Date 2023-12-16

Author Kathryn Leeming [aut], Guy Nason [aut], Matt Nunes [aut, cre], Marina Knight [ctb], James Wei [aut], Daniel Salnikov [aut], Mario Cortina Borja [ctb]

Maintainer Matt Nunes <nunesrpackages@gmail.com>

Description Simulation of, and fitting models for, Generalised Network Autoregressive (GNAR) time series models which take account of network structure, potentially with exogenous variables. Such models are described in Knight et al. (2020) <doi:10.18637/jss.v096.i05> and Nason and Wei (2021) <doi:10.1111/rssa.12875>. Diagnostic tools for GNAR(X) models can be found in Nason et al (2023) <arXiv:2312.00530>.

Depends R(>= 3.5.0)

Imports ggforce, ggplot2, ggpubr, grid, igraph, matrixcalc, rlang, stats, viridis, wordcloud

License GPL-2

Encoding UTF-8

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2023-12-18 13:40:08 UTC

Contents

`active_node_plot` *Produces an active node matrix heat-map.*

Description

Produces an active node matrix heat-map, which compares the local impact each node has on all the other ones (i.e., regressing j on i) once a model order has been chosen. The local relevance indes is $\text{local}(i,j) := \left(w_{ij} \sum_{k=1}^{p} |\hat{\beta}_{kr}| \right) \left\{ \sum_{l \in \mathcal{N}(i)} \sum_{r=1}^{r^*} \sum_{k=1}^{p} w_{il} |\hat{\beta}_{kr}| \right\}^{-1}$, which is closer to one the more relevant j is when forecasting i.

Usage

```
active_node_plot(vts, network, max_lag, r_stages)
```

Arguments

`vts`	Vector time series under study.
`network`	GNAR network object, which is the underlying network for the time series under study.
`max_lag`	Maximum lag of the fitted GNAR model - i.e., $\text{GNAR}(p, [s_1, \ldots, s_p])$.
`r_stages`	Neighbourhood regression oreder of the fitted GNAR model - i.e., $(s_1, \ldots, s_p)$.

Value

Produces the local influence matrix heat-map for a specific model order. Does not return any values.

Author(s)

Daniel Salnikov and Guy Nason

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
#
# Produces an active node heat-map matrix from a stationary GNAR(2, [2, 1]) simulation.
#
gnar_simulation <- GNARsim(n = 100, net=fiveNet,
 alphaParams = list(rep(0.25, 5), rep(0.12, 5)),
        betaParams = list(c(0.25, 0.13), c(0.20)), sigma=1)
#
# Active node plot
#
active_node_plot(gnar_simulation, fiveNet, 2, c(2, 1))
```

AIC.GNARfit *Akaike's Information Criterion for GNAR models*

Description

Function calculating AIC for GNARfit models.

Usage

```
## S3 method for class 'GNARfit'
AIC(object, ..., k=2)
```

Arguments

object	a GNARfit object, output from a GNARfit call.
...	additional arguments, not used here.
k	the penalty for the criterion, the default k = 2 is the standard AIC.

Details

Smaller AIC values correspond to better fit.

Value

A numeric value corresponding to the AIC (or other criterion if k is set to something other than 2). Note that the value returned is the "time-normalised" AIC for the GNAR model, and also removes any proportionality constants in the calculation.

Note

Occasionally it has been observed that when the forecast horizon has been set too high, this can result in NA AIC values. This can be resolved by reducing the forecast horizon. If package users find this problem persists or they experience any other unexpected behaviour, please contact the package maintainer.

Examples

```
#AIC for two different GNAR fits for fiveNet data
#GNAR(2,[1,1])
AIC(GNARfit())
#GNAR(2,[1,0])
AIC(GNARfit(betaOrder=c(1,0)))
```

`as.matrix.GNARnet` *Converts a GNAR networks into a weighted adjacency matrix*

Description

Takes an input GNARnet and neighbour stage and outputs the corresponding adjacency matrix.

Usage

```
## S3 method for class 'GNARnet'
as.matrix(x, stage=1, normalise=FALSE, ...)
```

Arguments

`x`	the network `GNARnet` object associated with the time series, containing a list with entries `$edges` and `$dist`.
`stage`	the neighbour set that the adjacency matrix is created for.
`normalise`	whether to normalise each to non-zero row to have sum one.
`...`	additional arguments, unused here.

Details

S3 method for class "GNARnet".

With normalisation this is a non-invertible transform. See NofNeighbours for neighbour set definition.

Value

`as.matrix` performed on a `GNARnet` returns a square matrix with the number of rows and columns equal to the length of the `$edges` list. Entry `i,j` of the matrix will be non-zero if node `j` is in the `stage` neighbour set of `i`.

Examples

```
#fiveNet as an adjacency matrix
as.matrix(fiveNet)
```

BIC.GNARfit *Bayesian Information Criterion for GNAR models*

Description

Function calculating BIC for GNARfit models.

Usage

```
## S3 method for class 'GNARfit'
BIC(object, ...)
```

Arguments

object	a GNARfit object, output from a GNARfit call.
...	additional arguments, not used here.

Details

Smaller BIC values correspond to better fit.

Value

A numeric value corresponding to the BIC. Note that this is the "time-normalised" value of the AIC for the GNAR model, and also removes any proportionality constants in the calculation.

Examples

```
#BIC for two different GNAR fits for fiveNet data
#GNAR(2,[1,1])
BIC(GNARfit())
#GNAR(2,[1,0])
BIC(GNARfit(betaOrder=c(1,0)))
```

coef.GNARfit *Function to return coefficients of GNARfit objects*

Description

coef.GNARfit returns the vector of coefficients from a GNARfit object.

Usage

```
## S3 method for class 'GNARfit'
coef(object,...)
```

Arguments

object	the output of a GNARfit call
...	additional arguments, unused here.

Details

S3 method for class "GNARfit".

Value

coef.GNARfit returns a vector of coefficient values.

Examples

```
#get the coefficients of the fiveNode data GNAR fit
coef(GNARfit())
```

---

corbit_plot *Corbit (correlation-orbit) plot, which aids model selection by visualising network autocorrelation and partial network autocorrelation.*

Description

Plots the GNAR network autcorrelation funciton for a choice of maximum lag and maximum r-stage depth in the network. Using the nacf function for network autocorrelation and pnacf for partial network autocorrelation.

Usage

```
corbit_plot(vts, net, max_lag, max_stage, weight_matrix,
    viridis_color_option="viridis", size_option="absolute_val",
    partial="no", wagner="no")
```

Arguments

vts	Vector time series observations for which one wishes to plot the network autocorrelation or partial network autocorrelation.
net	GNAR network object linked to the time series under study.
max_lag	Maximum lag the Corbit plot produces (i.e., number of time-steps considered for the network autocorrelaiton.)
max_stage	Maximum r-stage depth considered for the Corbit plot (i.e., the number of rings in the plot). Corresponds to the length of paths in the underlying network.
weight_matrix	A matrix which entries correspond to the weights between nodes. If this term is NULL, then this argument is equal weights between r-stage neighbours.

`viridis_color_option`
: Colour scale for the Corbit plot. The default option is `viridis`, each option is colout blind friendly; see `viridis` package.

`size_option`
: Point size scale for the Corbit plot. The default is the absolute value of the network autocorrelation function (i.e., `|nacf(h, r)|` or `|pnacf(h, r)|`). Alternate option is the coefficient of determination coming from a global-α model with fixed lag and stage.

`partial`
: Option for selecting between computing the network autocorrelation function or the partial network autocorrelation funciton. Default choice is network autocorrelation (i.e., partial="no"), change argument to "yes" for computing the partial network autocorrelation funciton.

`wagner`
: Choice for distinguishing between Corbit and Wagner plots, default is set to Corbit (inner function call). For producing Wagner plots one should use `wagner_plot`.

Details

Function calculates the network autocorrelation or the partial network autocorrelation values for a specific choice of maximum lag and r-stage depth, and produces the corresponing Corbit plot. Each point in the Corbit plot corresponds to the network autocorrelation `nacf(h, r)` at a h-lag and r-stage pair. The ring number starting from the inside corresponds to r-stage depth (path length), and the numbers on the outside ring indicate the time lag. The colour scale is based on the overall network autocorrelation values (i.e., the colour is set to highlight strong correlations).

Value

Produces the specified Corbit plot, does not return the network autocorrelaiton values.

Author(s)

Daniel Salnikov and Guy Nason.

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
## Not run:
#
# Simulate 100 observations from a stationary GNAR(2, [2, 1]), where fiveNet is
# the underlying network.
#
gnar_simulation <- GNARsim(n = 100, net=fiveNet, alphaParams = list(rep(0.25, 5), rep(0.12, 5)),
        betaParams = list(c(0.25, 0.13), c(0.20)), sigma=1)
# We produce the corresponding Corbit plots.
corbit_plot(gnar_simulation, fiveNet, 20, 3)
corbit_plot(gnar_simulation, fiveNet, 20, 3, partial = "yes")

# If the network object comes with its own weights, then these can be added by including the
```

```
# option weigh_matrix in the corbit call.
# corbit_plot(vts, net, max_lag, max_stage, weight_matrix = object_weights_matrix)

## End(Not run)
```

`fitted.GNARfit` *Function to return fitted values of GNARfit objects*

Description

`fitted.GNARfit` returns the fitted values of a GNARfit object as a matrix.

Usage

```
## S3 method for class 'GNARfit'
fitted(object,...)
```

Arguments

`object`	the output of a GNARfit call
`...`	additional arguments, unused here.

Details

S3 method for class "GNARfit".

Value

`fitted.GNARfit` returns a ts object of fitted values, with `t-alphaOrder` rows and `nnodes` columns.

Examples

```
#get the fitted values of the fiveNode GNAR fit
fitted(GNARfit())
```

fiveNode *Example Network Time Series*

Description

A multivariate time series `fiveVTS` and corresponding network `fiveNet`.

Usage

```
data("fiveNode")
```

Format

This dataset contains two R objects:
`fiveVTS` is a ts object with a matrix of 200 rows (t=200) and 5 columns (n=5) `fiveNet` is a GNAR-net object containing `$edges` and `$dist`.
`edges` is a list of length five, with `edges[[i]]` containing the vertices that node i is connected to.
`dist` is a list of length five, with `dist[[i]]` containing the length of the vertices that node i is connected to.

Examples

```
plot(fiveNet)
image(fiveVTS)
```

gdpVTS *Differenced GDP values for 35 countries*

Description

This dataset is from the OECD (OECD (2018), Quarterly GDP (indicator). <doi:10.1787/b86d1fc8-en> (Accessed on 29 January 2018)) and is differenced annual growth rate for 35 countries for 1962-2013.

Usage

```
gdpVTS
```

Format

`gdpVTS` is a ts object with a matrix of 52 rows (t=52) and 35 columns (n=35)

Examples

```
#Plot using 'ts' S3 function, can only plot up to 10 columns at once
plot(gdpVTS[,1:5])

#Plot as heatmap
image(gdpVTS)
```

`get_k_stages_adjacency_tensor` *Computes a list of r-stage adjacency matrices.*

Description

Computes a list of r-stage adjacency matrices, each matrix in the list inidicates whether or not nodes i and j are r-stage neighbours in the underlying network. Essentially $[\mathbf{S}_r]_{ij} = 1$ if and only if $d(i, j) = r$

Usage

```
get_k_stages_adjacency_tensor(St_1, r)
```

Arguments

`St_1`	One-stage adjacency matrix (i.e., the adjacency matrix of the underlying network).
`r`	Maximum r-stage for which one wishes to compute the r-stage adjacency matrix.

Value

List containing the adjacency matrices in ascending order.

`\eqn{ \{ \mathbf{S}_q \}_{q=1}^{q = r} }`
Each entry is the r-stage adjacency matrix at depth r.

Author(s)

Daniel Salnikov and Guy Nason

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
#
# Produce the r-stage adjacency tensors for the fiveNet network.
#
get_k_stages_adjacency_tensor(as.matrix(GNARtoigraph(fiveNet)), 3)
#
```

GNAR *GNAR package*

Description

A package to fit, predict, and simulate time series using the Generalised Network AutoRegressive (GNAR) model, potentially with exogenous variables. The main functions are GNARfit and GNARXfit, which fits the model to a time series and network(s), S3 method predict.GNARfit which predicts from a fitted GNAR model, and GNARsim which simulates from a GNAR model with specified parameters. For details of the model, see GNARfit. The package also contains an example network time series in data file fiveNode, with network fiveNet, and simulated time series fiveVTS.

References

Knight, M.I., Nunes, M.A. and Nason, G.P. (2015) Modelling, detrending and decorrelation of network time series. arXiv preprint.

Knight, M.I., Leeming, K., Nason, G.P. and Nunes, M. A. (2020) Generalised Network Autoregressive Processes and the GNAR package. *Journal of Statistical Software*, **96** (5), 1–36.

Nason G.P. and Wei J. (2022) Quantifying the economic response to COVID-19 mitigations and death rates via forecasting Purchasing Managers' Indices using Generalised Network Autoregressive models with exogenous variables. *Journal of the Royal Statistical Society Series A*, **185** (4), 1778–1792.

GNARdesign *Function to create the GNAR design matrix*

Description

Creates the design matrix necessary for fitting the GNAR model.

Usage

```
GNARdesign(vts = GNAR::fiveVTS, net = GNAR::fiveNet, alphaOrder = 2, betaOrder = c(1,1),
 fact.var = NULL, globalalpha=TRUE, tvnets=NULL, netsstart=NULL)
```

Arguments

`vts`	a matrix or ts object containing the multivariate time series to be modelled. The `i,j` entry of this matrix should be for time `i` and vertex/node `j`.
`net`	the (first) network associated with the time series, containing a list with entries `$edges` and `$dist`. This network should have the same number of nodes as the number of columns of the `vts` matrix.
`alphaOrder`	a non-negative integer specifying the maximum time-lag to model.

betaOrder	a vector of length alphaOrder specifying the maximum neighbour set to model at each of the time-lags.
fact.var	a vector of factors indicating which nodes belong to each set with different parameters to be fitted.
globalalpha	a TRUE/FALSE value indivating whether to use global alpha parameters.
tvnets	a list of additional networks. Currently only NULL (the static network case) is supported.
netsstart	a vector of times corresponding to the first time points for each network of tvnets. Currently only NULL (the static network case) is supported.

Value

GNARdesign	returns a matrix containing (t-alphaOrder)nnodes rows and a column for each parameter to be fitted. The columns are in time-lag order, eg for GNAR(2,[1,0]) the columns are alpha1, beta1.1, alpha2. When a factor variable is specified the columns are labelled with the factor.

Examples

```
#Design matrix to fit GNAR(2,[1,1]) to the fiveVTS data
GNARdesign()
```

GNARfit *Fitting function for GNAR models*

Description

Fits the GNAR model to the given inputs using GNARdesign and lm.

Usage

```
GNARfit(vts=GNAR::fiveVTS, net=GNAR::fiveNet, alphaOrder=2, betaOrder=c(1,1),
 fact.var=NULL, globalalpha=TRUE, tvnets=NULL, netsstart=NULL, ErrorIfNoNei=TRUE)
```

Arguments

vts	a matrix containing the multivariate time series to be modelled. The i,j entry of this matrix should be for time i and vertex/node j.
net	the (first) network associated with the time series, containing a list with entries $edges and $dist. This network should have the same number of nodes as the number of columns of the vts matrix.
alphaOrder	a non-negative integer specifying the maximum time-lag to model.
betaOrder	a vector of length alphaOrder specifying the maximum neighbour set to model at each of the time-lags.
fact.var	a vector of factors indicating which nodes belong to different sets with different parameters to be fitted.

globalalpha	a TRUE/FALSE value indivating whether to use global alpha parameters.
tvnets	a list of additional networks. Currently only NULL (the static network case) is supported.
netsstart	a vector of times corresponding to the first time points for each network of tvnets. Currently only NULL (the static network case) is supported.
ErrorIfNoNei	a TRUE/FALSE value indicating whether to stop the function call with an error if betaOrder specifies more neighbour sets than exist in the network. If FALSE the function will continue and some parameters will be NA.

Details

The GNAR model of order (p, S) is defined as

$$X_{i,t} = \sum_{j=1}^{p} \left(\alpha_{i,j} X_{i,t-j} + \sum_{c=1}^{C} \sum_{r=1}^{S_j} \beta_{j,r,c} \sum_{q \in N_t^{(r)}(i)} \omega_{i,q,c}^{(t)} X_{q,t-j} \right) + u_{i,t}$$

where p is the maximum time lag, $S = (S_1, ..., S_p)$ and S_j is the maximum stage of neighbour dependence for time lag j, $N_t^{(r)}(i)$ is the rth stage neighbour set of node i at time t, $\omega_{i,q,c}^{(t)}$ is the connection weight between node i and node q at time t if the path corresponds to covariate c. Here, we consider a sum from one to zero to be zero and $\{u_{i,t}\}$, are assumed to be independent and identically distributed at each node i, with mean zero and variance σ_i^2. Currently, only a single network GNAR model can be fitted. The connection weight, $\omega_{i,q,c}^{(t)}$, is the inverse of the distance between nodes i and q, normalised so that they sum to 1 for each i, t. See is.GNARnet for GNARnet object information and example construction.

Value

mod	the lm output from fitting the GNAR model.
y	the original response values, with NAs left in.
dd	the output of GNARdesign containing the design matrix, with NAs left in.
frbic	inputs to other GNAR functions.

References

Knight, M.I., Nunes, M.A. and Nason, G.P. Modelling, detrending and decorrelation of network time series. arXiv preprint.

Knight, M.I., Leeming, K., Nason, G.P. and Nunes, M. A. (2020) Generalised Network Autoregressive Processes and the GNAR package. *Journal of Statistical Software*, **96** (5), 1–36.

Examples

```
#Fit the GNAR(2,[1,1]) model to the fiveVTS data
GNARfit()

#Convert the residuals to matrix form
residToMat(GNARfit())$resid
```

GNARsim *Simulates a GNAR process*

Description

Simulates a GNAR process with Normally distributed innovations.

Usage

```
GNARsim(n=200, net=GNAR::fiveNet, alphaParams=list(c(rep(0.2,5))),
 betaParams=list(c(0.5)), sigma=1, tvnets=NULL, netsstart=NULL)
```

Arguments

n	time length of simulation.
net	network used for the GNAR simulation.
alphaParams	a list containing vectors of auto-regression parameters for each time-lag.
betaParams	a list of equal length as alphaParams containing the network-regression parameters for each time-lag.
sigma	the standard deviation for the innovations.
tvnets	Only NULL is currently supported.
netsstart	Only NULL is currently supported.

Details

Parameter lists should not be NULL, set unused parameters to be zero. See GNARfit for model description.

Value

GNARsim returns the multivariate time series as a ts object, with n rows and a column for each of the nodes in the network.

References

Knight, M.I., Nunes, M.A. and Nason, G.P. Modelling, detrending and decorrelation of network time series. arXiv preprint.

Knight, M.I., Leeming, K., Nason, G.P. and Nunes, M. A. (2020) Generalised Network Autoregressive Processes and the GNAR package. *Journal of Statistical Software*, **96** (5), 1–36.

Examples

```
#Simulate a GNAR(1,[1]) process with the fiveNet network
GNARsim()
```

GNARtoigraph *Converts a GNAR network to a weighted igraph object*

Description

Takes an input network and neighbour stage and returns it in igraph form.

Usage

```
GNARtoigraph(net=GNAR::fiveNet, stage=1, normalise=FALSE)
```

Arguments

net	a GNARnet object containing $edges and dist.
stage	the neighbour set that the adjacency matrix is created for.
normalise	whether to normalise each to non-zero row to have sum one.

Details

With normalisation this is a non-invertible transform. See NofNeighbours for neighbour set definition. See is.GNARnet for GNARnet object information and example construction.

Value

GNARtoigraph returns an 'igraph' object with weights as the inverse distances of the input network.

Examples

```
#fiveNet as an igraph object
GNARtoigraph()
```

GNARXdesign *Function to create the GNARX design matrix*

Description

Creates the design matrix necessary for fitting the GNAR model.

Usage

```
GNARXdesign(vts = GNAR::fiveVTS, net = GNAR::fiveNet, alphaOrder = 2, betaOrder = c(1,1),
 fact.var = NULL, globalalpha=TRUE, tvnets=NULL, netsstart=NULL, lambdaOrder=NULL,
 xvts=NULL)
```

Arguments

vts	a matrix or ts object containing the multivariate time series to be modelled. The i,j entry of this matrix should be for time i and vertex/node j.
net	the (first) network associated with the time series, containing a list with entries $edges and $dist. This network should have the same number of nodes as the number of columns of the vts matrix.
alphaOrder	a non-negative integer specifying the maximum time-lag to model.
betaOrder	a vector of length alphaOrder specifying the maximum neighbour set to model at each of the time-lags.
fact.var	a vector of factors indicating which nodes belong to each set with different parameters to be fitted.
globalalpha	a TRUE/FALSE value indivating whether to use global alpha parameters.
tvnets	a list of additional networks. Currently only NULL (the static network case) is supported.
netsstart	a vector of times corresponding to the first time points for each network of tvnets. Currently only NULL (the static network case) is supported.
lambdaOrder	a vector of the same length as xvts containing non-negative integers specifying the maximum time-lag of exogenous regressors to model. The hth element of the vector refers to the maximum lag of the hth exogenous regressor.
xvts	a list of matrices containing values of the exogenous regressors for each vertex/node. The i,j entry of the hth element of the list refers to the value of the hth exogenous regressor for time i and vertex/node j.

Value

GNARdesign	returns a matrix containing (t-alphaOrder)nnodes rows and a column for each parameter to be fitted. The columns are in time-lag order, eg for GNAR(2,[1,0]) the columns are alpha1, beta1.1, alpha2. When a factor variable is specified the columns are labelled with the factor. If exogenous regressors are included, the matrix takes the form given in Equation 10 of the supplement to the referenced paper.

References

Nason G.P. and Wei J. (2022) Quantifying the economic response to COVID-19 mitigations and death rates via forecasting Purchasing Managers' Indices using Generalised Network Autoregressive models with exogenous variables. *Journal of the Royal Statistical Society Series A*, **185**, 1778–1792.

Examples

```
set.seed(1)
n = 1000
xvts=list()
xvts[[1]] = matrix(rnorm(5*n, mean=0, sd=2), nrow=n, ncol=5)
xvts[[2]] = matrix(rnorm(5*n, mean=0, sd=2), nrow=n, ncol=5)
```

```
lambdaParams=list()
lambdaParams[[1]] = c(0.5, -0.5)
lambdaParams[[2]] = c(0.3, 0.1)

# Simulate the GNARX using the exogenous variables xvts with associated parameters lambdaParams

Y_data <- GNARXsim(n=n, net=GNAR::fiveNet, alphaParams=list(c(rep(0.2,5))), betaParams=list(c(0.5)),
                      sigma=1, xvts=xvts, lambdaParams=lambdaParams)

#Design matrix to fit GNARX(2,[1,1]) to the fiveVTS data

Xdesign <- GNARXdesign(vts = Y_data, net = GNAR::fiveNet, globalalpha = TRUE, alphaOrder = 1,
                betaOrder = 1, xvts = xvts, lambdaOrder = c(1,1))

Xdesign
```

GNARXfit *Fitting function for GNARX models*

Description

Fits the GNARX model to the given inputs using `GNARdesignX` and `lm`.

Usage

```
GNARXfit(vts=GNAR::fiveVTS, net=GNAR::fiveNet, alphaOrder=2, betaOrder=c(1,1),
 fact.var=NULL, globalalpha=TRUE, tvnets=NULL, netsstart=NULL, ErrorIfNoNei=TRUE,
 xvts=NULL, lambdaOrder=NULL)
```

Arguments

`vts`	a matrix containing the multivariate time series to be modelled. The `i,j` entry of this matrix should be for time `i` and vertex/node `j`.
`net`	the (first) network associated with the time series, containing a list with entries `$edges` and `$dist`. This network should have the same number of nodes as the number of columns of the `vts` matrix.
`alphaOrder`	a non-negative integer specifying the maximum time-lag to model.
`betaOrder`	a vector of length `alphaOrder` specifying the maximum neighbour set to model at each of the time-lags.
`fact.var`	a vector of factors indicating which nodes belong to different sets with different parameters to be fitted.
`globalalpha`	a TRUE/FALSE value indivating whether to use global alpha parameters.
`tvnets`	a list of additional networks. Currently only NULL (the static network case) is supported.
`netsstart`	a vector of times corresponding to the first time points for each network of `tvnets`. Currently only NULL (the static network case) is supported.

ErrorIfNoNei	a TRUE/FALSE value indicating whether to stop the function call with an error if betaOrder specifies more neighbour sets than exist in the network. If FALSE the function will continue and some parameters will be NA.
xvts	a list of matrices containing values of the exogenous regressors for each vertex/node. The i,j entry of the hth element of the list refers to the value of the hth exogenous regressor for time i and vertex/node j.
lambdaOrder	a vector of the same length as xvts containing non-negative integers specifying the maximum time-lag of exogenous regressors to model. The hth element of the vector refers to the maximum lag of the hth exogenous regressor.

Details

The GNAR model of order (p, S) is defined as

$$X_{i,t} = \sum_{j=1}^{p} \left(\alpha_{i,j} X_{i,t-j} + \sum_{c=1}^{C} \sum_{r=1}^{S_j} \beta_{j,r,c} \sum_{q \in N_t^{(r)}(i)} \omega_{i,q,c}^{(t)} X_{q,t-j} \right) + u_{i,t}$$

where p is the maximum time lag, $S = (S_1, ..., S_p)$ and S_j is the maximum stage of neighbour dependence for time lag j, $N_t^{(r)}(i)$ is the rth stage neighbour set of node i at time t, $\omega_{i,q,c}^{(t)}$ is the connection weight between node i and node q at time t if the path corresponds to covariate c. Here, we consider a sum from one to zero to be zero and $\{u_{i,t}\}$, are assumed to be independent and identically distributed at each node i, with mean zero and variance σ_i^2. Currently, only a single network GNAR model can be fitted. The connection weight, $\omega_{i,q,c}^{(t)}$, is the inverse of the distance between nodes i and q, normalised so that they sum to 1 for each i,t. See is.GNARnet for GNARnet object information and example construction.

A GNARX process of order p, neighbourhood order vector s of length p and H node-specific time series exogenous variables with order vector p', denoted GNARX (p, s, p'), is given by

$$Y_{i,t} = \sum_{j=1}^{p} \left(\alpha_{i,j} Y_{i,t-j} + \sum_{r=1}^{s_j} \beta_{j,r} \sum_{q \in \mathcal{N}^{(r)}(i)} \omega_{i,q} Y_{q,t-j} \right) + \sum_{h=1}^{H} \sum_{j'=0}^{p'_h} \lambda_{h,j'} X_{h,i,t-j'} + u_{i,t},$$

where $u_{i,t}$ are assumed to be set of mutually uncorrelated random variables with mean zero and variance of σ^2.

Value

mod	the lm output from fitting the GNAR model.
y	the original response values, with NAs left in.
dd	the output of GNARdesign containing the design matrix, with NAs left in.
frbic	inputs to other GNAR functions.

References

Knight, M.I., Nunes, M.A. and Nason, G.P. Modelling, detrending and decorrelation of network time series. arXiv preprint.

Knight, M.I., Leeming, K., Nason, G.P. and Nunes, M. A. (2020) Generalised Network Autoregressive Processes and the GNAR package. *Journal of Statistical Software*, **96** (5), 1–36.

Nason G.P. and Wei J. (2022) Quantifying the economic response to COVID-19 mitigations and death rates via forecasting Purchasing Managers' Indices using Generalised Network Autoregressive models with exogenous variables. *Journal of the Royal Statistical Society Series A*, **185**, 1778–1792.

Examples

```
set.seed(1)
n = 1000
xvts=list()
xvts[[1]] = matrix(rnorm(5*n, mean=0, sd=2), nrow=n, ncol=5)
xvts[[2]] = matrix(rnorm(5*n, mean=0, sd=2), nrow=n, ncol=5)
lambdaParams=list()
lambdaParams[[1]] = c(0.5, -0.5)
lambdaParams[[2]] = c(0.3, 0.1)

# Simulate the GNARX using the exogenous variables xvts with associated parameters lambdaParams

Y_data <- GNARXsim(n=n, net=GNAR::fiveNet, alphaParams=list(c(rep(0.2,5))), betaParams=list(c(0.5)),
                       sigma=1, xvts=xvts, lambdaParams=lambdaParams)

# fit a GNARX to the model

model <- GNARXfit(vts = Y_data, net = GNAR::fiveNet,globalalpha = TRUE, alphaOrder = 1,
                betaOrder = 1, xvts = xvts, lambdaOrder = c(1,1))

# look at the residuals
residToMat(model)$resid
```

GNARXsim *Simulates a GNARX process*

Description

Simulates a GNAR process with Normally distributed innovations.

Usage

```
GNARXsim(n=200, net=GNAR::fiveNet, alphaParams=list(c(rep(0.2,5))),
 betaParams=list(c(0.5)), sigma=1, tvnets=NULL, netsstart=NULL, xvts=NULL,
 lambdaParams=NULL)
```

Arguments

n	time length of simulation.
net	network used for the GNAR simulation.
alphaParams	a list containing vectors of auto-regression parameters for each time-lag.
betaParams	a list of equal length as alphaParams containing the network-regression parameters for each time-lag.
sigma	the standard deviation for the innovations.
tvnets	a list of additional networks. Currently only NULL (the static network case) is supported.
netsstart	a vector of times corresponding to the first time points for each network of tvnets. Currently only NULL (the static network case) is supported.
xvts	a list of matrices containing values of the exogenous regressors for each vertex/node. The i,j entry of the hth element of the list refers to the value of the hth exogenous regressor for time i and vertex/node j.
lambdaParams	a list containing vectors of parameters associated to effect of the exogenous regressor variables for each time-lag.

Details

Parameter lists should not be NULL, set unused parameters to be zero. See GNARXfit for model description.

Value

GNARXsim returns the multivariate time series as a ts object, with n rows and a column for each of the nodes in the network.

References

Knight, M.I., Nunes, M.A. and Nason, G.P. Modelling, detrending and decorrelation of network time series. arXiv preprint.

Knight, M.I., Leeming, K., Nason, G.P. and Nunes, M. A. (2020) Generalised Network Autoregressive Processes and the GNAR package. *Journal of Statistical Software*, **96** (5), 1–36.

Nason G.P. and Wei J. (2022) Quantifying the economic response to COVID-19 mitigations and death rates via forecasting Purchasing Managers' Indices using Generalised Network Autoregressive models with exogenous variables. *Journal of the Royal Statistical Society Series A*, **185**, 1778–1792.

Examples

```
#Simulate a GNARX process with the fiveNet network

set.seed(1)
n = 1000
xvts=list()
```

```
xvts[[1]] = matrix(rnorm(5*n, mean=0, sd=2), nrow=n, ncol=5)
xvts[[2]] = matrix(rnorm(5*n, mean=0, sd=2), nrow=n, ncol=5)
lambdaParams=list()
lambdaParams[[1]] = c(0.5, -0.5)
lambdaParams[[2]] = c(0.3, 0.1)

# Simulate the GNARX using the exogenous variables xvts with associated parameters lambdaParams

Y_data <- GNARXsim(n=n, net=GNAR::fiveNet, alphaParams=list(c(rep(0.2,5))), betaParams=list(c(0.5)),
                      sigma=1, xvts=xvts, lambdaParams=lambdaParams)

# now try to refit the model

model <- GNARXfit(vts = Y_data, net = GNAR::fiveNet,globalalpha = TRUE, alphaOrder = 1,
  betaOrder = 1, xvts = xvts, lambdaOrder = c(1,1))

model
```

`igraphtoGNAR` *Converts an igraph to GNAR network*

Description

Converts an 'igraph' to the `GNARnet` form for use as an input to GNAR functions.

Usage

```
igraphtoGNAR(ig)
```

Arguments

`ig`	an 'igraph' object.

Details

The values in the `$dist` list are the reciprocal of the values from the weighted adjacency matrix.

Value

`igraphtoGNAR` returns a `GNARnet`: a list with elements `$edges` and `$dist`.

Examples

```
#Convert fiveNet to igraph and back again
igraphtoGNAR(GNARtoigraph(fiveNet))
```

is.GNARfit — *Function to check GNARfit objects*

Description

`is.GNARfit` returns either TRUE or FALSE according to a series of GNARfit checks.

Usage

```
is.GNARfit(x)
```

Arguments

`x`	the object to be tested

Details

The `is.GNARfit` function checks whether the object passes a series of tests that correspond to it being the output of GNARfit:

- Is it a list containing `$mod` and `$frbic`
- Does it contain either `$y` and `$dd` or `$ys` and `$ds`
- Is `$mod` a lm object
- Does `$frbic` have the components to calculate the BIC with BIC.GNARfit

Value

`is.GNARfit` returns `TRUE` or `FALSE` corresponding to passing the above tests.

Examples

```
#check that the example fit meets the criteria above
is.GNARfit(GNARfit())
```

is.GNARnet — *Functions to check and create GNARnet objects*

Description

`is.GNARnet` returns either TRUE or FALSE according to a series of GNARnet checks. `as.GNARnet` returns a GNARnet object from an input weights matrix, 'igraph' object, or a GNARnet without assigned class.

Usage

```
is.GNARnet(x)
as.GNARnet(x)
```

Arguments

x	the network to be tested or object to be converted

Details

The `is.GNARnet` function checks whether the network passes a series of tests:

- Is it a list containing $edges and $dist
- Are the $edges and $dist lists of the same length
- Are each of the elements of $edges the same length as the corresponding $dist element
- Do the edges only contain valid entries, 1,...,nnodes (or `NULL`)
- Is it labelled as `GNARnet` class
- Are no duplicate edges present
- Are all distances positive
- Are there no self-loops in the network

The `as.GNARnet` function converts igraph objects to GNARnet form, other possible inputs are adjacency matrices, and lists with `$edges` and `$dist` entries of the correct form.

Value

`is.GNARnet` returns `TRUE` or `FALSE` corresponding to passing the above tests. `as.GNARnet` returns a `GNARnet` object.

Examples

```
#check that the example network meets the criteria above
is.GNARnet(fiveNet)

#convert to igraph and back again
as.GNARnet(GNARtoigraph(fiveNet))

#generate a new network with three nodes
#edges 1->2, 2->1, 2->3
#dist 1, 2, 1
#note 1->2 and 2->1 are of different lengths
threeEdge <- list(c(2), c(1,3), NULL)
threeDist <- list(c(1), c(2,1), NULL)
threeNet <- list(edges=threeEdge, dist=threeDist)
#check if this is a GNARnet
is.GNARnet(threeNet)
#use as.GNARnet to change the class
threeNet <- as.GNARnet(threeNet)
#check if this is a GNARnet now
is.GNARnet(threeNet)
```

`local_relevance_plot` *Produces a local neighbourhood relevance plot based on the distances in the underlying network.*

Description

Produces a local neighbourhood relevance plot based on the distances in the underlying network. The heat-map matrix should reflect clusters if a GNAR model is valid. The size of the clusters depends on the maximum r-stage depth for neighbourhood regression, as r^* gets larger, the clusters grow or intersect and cover more nodes. The relative strength of conditionally correlated nodes is $\text{rscc}(i,j) := \{d(i,j)\}^{-1}\mathbb{I}\{d(i,j) \leq r^*\} + \{2d(i,j)\}^{-1}\mathbb{I}\{r^* < d(i,j) \leq 2r^*\}$.

Usage

```
local_relevance_plot(network, r_star)
cross_correlation_plot(h, vts)
```

Arguments

`network`	GNAR network object, which is the underlying network for the time series under study.
`r_star`	Maximum active r-stage depth for neighbourhood regression.
`h`	The lag in the cross correlation plot.
`vts`	The vector time series to compute the cross correlation plot on.

Value

Produces the local relevance plot. Does not return any values.

Author(s)

Daniel Salnikov and Guy Nason

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
#
# Produces a local relevance plot, which is a heat-map matrix from a stationary
# GNAR(1, [1]) simulation.
#
gnar_simulation <- GNARsim(n = 100, net=fiveNet, alphaParams = list(rep(0.35, 5)),
        betaParams = list(c(0.25)), sigma=1)
# Active node plot
```

```
local_relevance_plot(fiveNet, 1)
# Compare to the cross-correlation plot at one-lag
cross_correlation_plot(1, gnar_simulation)
```

`logLik.GNARfit` *Log-likelihood method for GNARfit objects*

Description

Returns the log-likelihood for a `GNARfit` object.

Usage

```
## S3 method for class 'GNARfit'
logLik(object,...)
```

Arguments

`object`	A `GNARfit` object generated by a GNARfit call.
`...`	Optional additional arguments, not used here.

Details

S3 method for the GNARfit class. The function returns the value of

$$-TN/2\log(2\pi) - T/2\log(|\Sigma|) - 1/2trace(E\Sigma^{-1}E'),$$

where T is the time length of the observations, N is the number of nodes, $\Sigma = EE'/T$ is the estimated covariance matrix and E is the matrix of residuals.

Value

A `logLik` object.

Examples

```
#calculate log-likelihood for fiveNode data
#global alphas
logLik(GNARfit())
#individual alphas
logLik(GNARfit(globalalpha=FALSE))
```

logMVbedMVC.vts *Number of COVID19 admission to mechanical ventilation beds*

Description

This matrix contains a multivariate/vector time series that counts the number of daily admissions to mechanical ventilation beds in one of 140 NHS Trusts from 2nd April 2020 to 27th June 2021.

Format

The dimension of the matrix is 452x140. The dates and Trust ID codes are stored in the dimnames (first and second) respectively.

SEE ALSO

corbit_plot,NHSTrustMVCAug120.net

Author(s)

Daniel Salnikov and Guy Nason

Source

UK Coronavirus website https://coronavirus.data.gov.uk

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
## Not run:
data(logMVbedMVC.vts)
data(NHSTrustMVCAug120.net)

#
# Do a corbit plot with this data, with only three lags and one stage
#
# Note, normally max_lag and max_stage would be higher than this, the
# values are artificially small here, as otherwise the run-time restrictions
# for CRAN packaging might be exceeded.

corbit_plot(vts=logMVbedMVC.vts, net=NHSTrustMVCAug120.net, max_lag=3, max_stage=1)

## End(Not run)
```

matrixtoGNAR *Converts an adjacency matrix to GNAR network*

Description

Converts an adjacency matrix to the GNARnet form for use as an input to GNAR functions.

Usage

```
matrixtoGNAR(input.mat)
```

Arguments

input.mat	an adjacency matrix whose dimension equals the number of nodes in the resulting network.

Details

The values in the $dist list are the reciprocal of the values from the weighted adjacency matrix. Any self-loops (diagonal entries) and negatively weighted edges are removed.

Value

matrixtoGNAR returns a GNARnet list with elements $edges and $dist.

Examples

```
#Convert fiveNet to an adjacency matrix and back again
matrixtoGNAR(as.matrix(fiveNet))
```

na.row *Identifies which rows of a matrix have NAs*

Description

Returns a vector with elements TRUE/FALSE identifying which rows contain NA elements.

Usage

```
na.row(mat)
```

Arguments

mat	a matrix object.

Details

This function is used in the unstacking of residuals into a residual matrix and replacing NAs where they were previously present.

Value

`na.row` returns a vector of length equal to the number of rows in `mat`. Each element is either TRUE or FALSE.

Examples

```
#Check if there are and NAs in fiveVTS
na.row(fiveVTS)
```

---

nacf *Computes the Network Autocorrelation Function (NACF)*

Description

Computes the NACF for a choice of lag h and r-stage depth r, the NACF is given by $\text{nacf}(h, r) = \frac{\sum_{t=1}^{T-h}(\boldsymbol{X}_{t+h}-\overline{\boldsymbol{X}})^T\left(\mathbf{W}\odot\mathbf{S}_r+\mathbf{I_d}\right)(\boldsymbol{X}_t-\overline{\boldsymbol{X}})}{\sum_{t=1}^{T}(\boldsymbol{X}_t-\overline{\boldsymbol{X}})^T\left\{\left(1+\lambda\right)\mathbf{I_d}\right\}(\boldsymbol{X}_t-\overline{\boldsymbol{X}})}$, where $\lambda = \left[\max_{j=1,\ldots,d}\left\{\sum_{i=1}^{d}[(\mathbf{W}\odot\mathbf{W})]_{ij}\right\}\right]^{\frac{1}{2}}$ is the auto-covariance bound of the GNAR process.

Usage

```
nacf(h, s, weight_matrix, stages_tensor, nts_data)
```

Arguments

`h`	Lag (i.e., time-steps behind) at which the NACF is computed.
`s`	r-stage depth at which the NACF is computed (i.e., shortest distance between nodes).
`weight_matrix`	Weight matrix of the GNAR process, each entry corresponds to the weight between two nodes; see `weights_matrix`
`stages_tensor`	List of r-stage adjacency matrices $\mathbf{S}_r$, the order is ascending.
`nts_data`	Network time series observations, the number of rows is equal to the number of time steps, and the number of columns is equal to the number of series (variables).

Details

If the network time series contains missing values, then the weights matrix and λ are adjusted, so that missing values do not contribute to the network autocorrelation. This is done by setting to zero the weights which correspond to a missing value and computing the new weight matrix and λ value.

Value

Returns a number between -1 and 1, which measures the linear correlation between lagged network observations - i.e., $\mathrm{nacf}(h, r)$.

Author(s)

Daniel Salnikov and Guy Nason.

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
#
# Compute the NACF with respect to a stationary GNAR simulation
#
gnar_simulation <- GNARsim(n = 100, net=fiveNet, alphaParams = list(rep(0.35, 5)),
                           betaParams = list(c(0.25)), sigma=1)
W = weights_matrix(fiveNet)
stages_list = get_k_stages_adjacency_tensor(as.matrix(GNARtoigraph(fiveNet)), 3)
# NACF
nacf(3, 1, W, stages_list, gnar_simulation)
```

NHSTrustMVCAug120.net *Constructed network linking 140 NHS Trusts in England and Wales*

Description

This matrix contains a multivariate/vector time series that counts the number of daily admissions to mechanical ventilation beds in one of 140 NHS Trusts from 2nd April 2020 to 27th June 2021.

Format

An object of class GNARnet from the GNAR package

SEE ALSO

corbit_plot,NHSTrustMVCAug120.net

Author(s)

Daniel Salnikov and Guy Nason

Source

UK Coronavirus website https://coronavirus.data.gov.uk

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
## Not run:
data(logMVbedMVC.vts)
data(NHSTrustMVCAug120.net)

#
# Plot the network and see what it is like
#
plot(NHSTrustMVCAug120.net)

#
# Do a corbit plot with this data, with only three lags and one stage
#
# Note, normally max_lag and max_stage would be higher than this, the
# values are artificially small here, as otherwise the run-time restrictions
# for CRAN packaging might be exceeded.

corbit_plot(vts=logMVbedMVC.vts, net=NHSTrustMVCAug120.net, max_lag=3, max_stage=1)

## End(Not run)
```

`nobs.GNARfit` *Function to return the number of observations input to GNARfit objects*

Description

`nobs` returns the number of obervations (T) of the input multivariate time series in the `GNARfit` function.

Usage

```
## S3 method for class 'GNARfit'
nobs(object,...)
```

Arguments

`object`	the output of a GNARfit or GNARpredict call
`...`	additional arguments, unused here.

Details

S3 method for class "GNARfit".

Value

An integer specifying the number of rows in the input `vts` to the GNARfit function.

Examples

```
#observations of example fiveVTS
nobs(GNARfit())
#check this is the same as number of rows in fiveVTS
all.equal(nobs(GNARfit()), nrow(fiveVTS))
```

`node_relevance_plot` *Produces a node relevance plot, which compares the impact each node has on the network autocorrelation once a model order has been chosen.*

Description

Produces a node relevance plot based on the node relevance index $\text{globindex}(X_{i,t}) := \left(\sum_{j=1}^{d} [\mathbf{W} \odot \mathbf{S}]_{ji} \right) \left\{ \max_{l \in \mathcal{K}} \left(\sum_{j=1}^{d} [\mathbf{W} \odot \mathbf{S}]_{jl} \right) \right\}^{-1}$, which computes the ratio between nodes i column sum for nodes in neighbourhood regressions. Nodes are ordered according to the relative contribution eahc has to the autocovariance. The nodes are ordered in ascending order.

Usage

```
node_relevance_plot(network, r_star, node_names, node_label_size = 2)
```

Arguments

`network`	GNAR network object, which is the underlying network for the time series under study.
`r_star`	Maximum active r-stage depth for neighbourhood regression.
`node_names`	Names corresponding to each, this makes identifying nodes in the plot easier. If this argument is NULL, then the plot links to each node a number.
`node_label_size`	Text size when producing the plot. Default is 2, however, depending on the number of nodes it might be necessary to adjust the size.

Value

Data Frame consisting of two variable, the node name and the node relevance value.

Author(s)

Daniel Salnikov and Guy Nason.

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
#
# Produces a node relevance plot with respect to a stationary GNAR process
# with underlying network fiveNet
#
# GNAR simulation
gnar_simulation <- GNARsim(n = 100, net=fiveNet, alphaParams = list(rep(0.25, 5), rep(0.12, 5)),
        betaParams = list(c(0.25, 0.13), c(0.20)), sigma=1)
# Node relevance plot without names
node_relevance_plot(network = fiveNet, r_star = 2, node_label_size = 10)
#
# Node relevance plot with names
#
node_relevance_plot(network = fiveNet, r_star = 2, node_names = c("A", "B", "C", "D", "E"),
 node_label_size = 10)
```

`NofNeighbours` *Calculates stage-neighbours of a network*

Description

Calculates neighbour sets of a particular node in the network and their distances.

Usage

```
NofNeighbours(node=1, stage=2, net=GNAR::fiveNet)
```

Arguments

`node`	is an integer specifying which node to calculate the neighbours of.
`stage`	is an integer specifying the maximum neighbour-stage to calculate to.
`net`	a `GNARnet` object with edge list and distance list.

Details

Note that the distances are calculated as the sum along the shortest path; do not use this with a weights (rather than distance) list. Stage-`r` neighbours of node `i` are denoted $N^(r)(i)$, and are nodes that are `r` edges (but no fewer) away from `i`. Hence stage-1 neighbours are the immediate neighbours, stage-2 neighbours are the neighbours of neighbours and so on.

Value

edges	is a list of length stage, where edges[[i]] is a vector containing the nodes that are stage-i neighbours of node.
dist	is a list of length stage, where dist[[i]] is a vector containing the distances from node to its stage-i neighbours, with ordering as in edges[[i]].

Examples

```
#First and second stage neighbours of node 1 in fiveNet
NofNeighbours()
```

---

plot.GNARnet *Plot function for GNAR networks*

Description

Plots a GNAR network using the 'igraph' package.

Usage

```
## S3 method for class 'GNARnet'
plot(x, ...)
```

Arguments

x	the networkGNARnet object associated with the time series, containing a list with entries $edges and $dist.
...	additional arguments for the igraph plotting function, see plot.igraph.

Details

S3 method for class "GNARnet".

Examples

```
#Plot fiveNet
plot(fiveNet)
```

pnacf *Computes the Partial Network Autocorrelation Function (PNACF)*

Description

Computes the PNACF for a choice of lag h and r-stage depth r, the PNACF is given by $\text{pnacf}(h, r) = \frac{\sum_{t=1}^{T-h}(\hat{\boldsymbol{u}}_{t+h}-\overline{\boldsymbol{u}})^T\left(\mathbf{W}\odot\mathbf{S}_r+\mathbf{I_d}\right)(\hat{\boldsymbol{u}}_t-\overline{\boldsymbol{u}})}{\sum_{t=1}^{T}(\hat{\boldsymbol{u}}_t-\overline{\boldsymbol{u}})^T\left\{\left(1+\lambda\right)\mathbf{I_d}\right\}(\hat{\boldsymbol{u}}_t-\overline{\boldsymbol{u}})}$, where $\hat{\boldsymbol{X}}_t^{h-1,r-1} = \sum_{k=1}^{h-1}(\hat{\alpha}_k\boldsymbol{X}_{t-k}+\sum_{s=1}^{r-1}\hat{\beta}_{ks}\boldsymbol{Z}_{t-k}^s)$, $\hat{\boldsymbol{u}}_{t+h} = \boldsymbol{X}_{t+h}-\hat{\boldsymbol{X}}_{t+h}^{h-1,r-1}$, and $\hat{\boldsymbol{u}}_t = \boldsymbol{X}_t-\hat{\boldsymbol{X}}_t^{h-1,r-1}$ are the empirical residuals corresponding to GNAR(h -1, [r-1, ..., r - 1]) fits, λ is the same as for the NACF; see nacf, and $\overline{\boldsymbol{u}}$ is the mean of the fitted residuals

Usage

```
pnacf(h, s,  weight_matrix, stages_tensor, nts_data)
```

Arguments

h	Lag (i.e., time-steps behind) at which the NACF is computed.
s	r-stage depth at which the NACF is computed (i.e., shortest distance between nodes).
weight_matrix	Weight matrix of the GNAR process, each entry corresponds to the weight between two nodes; see weights_matrix
stages_tensor	List of r-stage adjacency matrices $\mathbf{S}_r$, the order is ascending.
nts_data	Network time series observations, the number of rows is equal to the number of time steps, and the number of columns is equal to the number of series (variables).

Value

If the network time series contains missing values, then the weights matrix and λ are adjusted, so that missing values do not contribute to the partial network autocorrelation. This is done by setting to zero the weights which correspond to a missing value and computing the new weight matrix and λ value.

Author(s)

Daniel Salnikov and Guy Nason

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
#
# Compute the PNACF with respect to a stationary GNAR simulation
#
gnar_simulation <- GNARsim(n = 100, net=fiveNet, alphaParams = list(rep(0.35, 5)),
                           betaParams = list(c(0.25)), sigma=1)
W = weights_matrix(fiveNet)
stages_list = get_k_stages_adjacency_tensor(as.matrix(GNARtoigraph(fiveNet)), 3)
# PNACF
pnacf(3, 1, W, stages_list, gnar_simulation)
```

`predict.GNARfit` *Prediction of a GNARfit object*

Description

Predicts future observations from a GNARfit object, based on the fitted GNAR model.

Usage

```
## S3 method for class 'GNARfit'
predict(object, n.ahead=1, ...)
```

Arguments

`object`	the output of a GNARfit call
`n.ahead`	the time length of the predictions
`...`	further arguments passed to the simulate.GNARfit function, such as `seed`

Details

S3 method for class "GNARfit". This function calls simulate.GNARfit.

Value

A multivariate time series of dimension `n.ahead` x nnodes.

Examples

```
#simulate 5 future observations from fiveVTS
predict(GNARfit(), n.ahead=5)
```

print.GNARfit *Function to print the model and coefficients of GNARfit objects*

Description

`print.GNARfit` prints model, call, and coefficients of a GNARfit object.

Usage

```
## S3 method for class 'GNARfit'
print(x,...)
```

Arguments

`x`	the output of a GNARfit call
`...`	additional arguments, unused here.

Details

S3 method for class "GNARfit".

Examples

```
#print the information of the fiveNode GNAR fit
print(GNARfit())
```

print.GNARnet *Print function for GNAR networks*

Description

Prints information about a GNAR network.

Usage

```
## S3 method for class 'GNARnet'
print(x, ...)
```

Arguments

`x`	the network `GNARnet` object associated with the time series, containing a list with entries `$edges` and `$dist`.
`...`	additional arguments, unused here.

Details

S3 method for class "GNARnet".

Examples

```
#print fiveNet information
print(fiveNet)
```

`residToMat` *Converts the output of a GNARfit call to fitted and residual value matrices*

Description

Unstacks the entries of the GNARfit fitted and residual values to return matrices of a similar form to the multivariate time series input.

Usage

```
residToMat(GNARobj=GNARfit(), nnodes=5)
```

Arguments

`GNARobj`	the output from the GNARfit function
`nnodes`	the number of nodes in the original network time series

Details

This function also replaces the NAs that were removed in fitting.

Value

`resid`	is the matrix of residual values, with `t-alphaOrder` rows and `nnodes` columns.
`fit`	is the matrix of fitted values, with `t-alphaOrder` rows and `nnodes` columns.

Examples

```
#Get residual and fitted matrices from GNARpredict fit of fiveVTS
residToMat()
```

residuals.GNARfit *Function to return residuals of GNARfit objects*

Description

`residuals.GNARfit` returns the residuals of a GNARfit object as a matrix.

Usage

```
## S3 method for class 'GNARfit'
residuals(object,...)
```

Arguments

`object`	the output of a GNARfit call
`...`	additional arguments, unused here.

Details

The function first checks if the object is of GNARfit class, then uses residToMat to return the residuals.

Value

`residuals.GNARfit` returns a 'ts' object of residuals, with `t-alphaOrder` rows and `nnodes` columns.

Examples

```
#get the residuals of the fiveNode GNAR fit
residuals(GNARfit())
```

seed.nos *Vector of seed numbers*

Description

Seed numbers for reproducible random graphs.

Usage

```
seed.nos
```

Format

`seed.nos` is a vector of length 10,000 containing integers.

Examples

```
g <- seedToNet(seed.nos[1], nnodes=35, graph.prob=0.15)
plot(g, vertex.label=colnames(gdpVTS), vertex.size=0)
```

`seedToNet` *Produces a random network from a seed value*

Description

Produces a reproducible undirected Erdos-Reyni random network using a particular seed value.

Usage

```
seedToNet(seed.no, nnodes=34, graph.prob=0.5)
```

Arguments

`seed.no`	a valid number to set the seed to.
`nnodes`	the number of nodes in the produced network.
`graph.prob`	the probability that each pair of nodes is connected.

Details

`graph.prob` effectively controls the sparsity of the network. All distances are set to 1.

Value

A GNARnet object.

Examples

```
#Generate the random graph from seed 10, with 5 nodes and connection prob 0.5
seedToNet(10,nnodes=5,graph.prob=0.5)
```

`simulate.GNARfit` *Function to simulate from a GNARfit object*

Description

Simulates from a GNARfit object, either creating a new series or future observations of the original series based upon the fitted GNAR model.

Usage

```
## S3 method for class 'GNARfit'
simulate(object, nsim=object$frbic$time.in, seed=NULL,
                              future=TRUE, set.noise=NULL, allcoefs=FALSE, ...)
```

Arguments

`object`	the output of a GNARfit call
`nsim`	the time length of the simulations
`seed`	either `NULL`, or a value to set the seed to
`future`	whether the simulations follow on from the original time series (`TRUE`), or if `FALSE` the simulations are a new series.
`set.noise`	a value to set the standard deviation of the noise to, or if `NULL`, the estimated standard deviation from the input series will be used.
`allcoefs`	if `TRUE`, all fitted coefficients will be used, if `FALSE` only the significant (p-val < 0.05) coefficients will be used.
`...`	additional arguments, unused here.

Details

S3 method for class "GNARfit".

Value

A multivariate time series of dimension `nsim` x nnodes.

Examples

```
#simulate 5 future observations from fiveVTS
simulate(GNARfit(), nsim=5)
```

summary.GNARfit *Returns model summary for a GNAR model fit*

Description

Returns the summary of a GNARfit object, including BIC.

Usage

```
## S3 method for class 'GNARfit'
summary(object, ...)
```

Arguments

object	output of a GNARfit call.
...	additional arguments, unused here.

Details

The output is the summary of the fit using summary.lm, and BIC calculated using BIC.GNARfit.

Value

summary.GNARfit prints the model summary and the value of the BIC.

Examples

```
#summary for the GNAR(2,[1,1]) model using GNARfit on fiveVTS
summary(GNARfit())
```

summary.GNARnet *Summary function for GNAR networks*

Description

Prints brief information about a GNAR network.

Usage

```
## S3 method for class 'GNARnet'
summary(object, ...)
```

Arguments

object	the networkGNARnet object associated with the time series, containing a list with entries $edges and $dist.
...	additional arguments, unused here.

Details

S3 method for class "GNARnet".

Examples

```
#print fiveNet summary information
summary(fiveNet)
```

`vcov.GNARfit` *Calculate variance-covariance matrix for a itted GNARfit object*

Description

Returns the varaince-covariance matrix of the parameters of a GNARfit object.

Usage

```
## S3 method for class 'GNARfit'
vcov(object,...)
```

Arguments

`object`	a GNARfit object, the output from a GNARfit call.
`...`	further arguments passed to vcov.

Details

S3 method for class "GNARfit".

Value

A matrix of estimated covariances between the parameter estimates, this is calculated using vcov for lm objects.

Examples

```
#covariance matrix of fiveNode fit
vcov(GNARfit())
```

vswind *Wind Speed example network time series*

Description

A suite of data objects concerning wind speed analysis. The dataset contains a multivariate time series of wind speeds, two network descriptions, a vector of names for weather stations, and the coordinates of the weather stations.

Usage

```
data("vswind")
```

Format

This dataset contains six R objects:
`vswindts` is a ts object with a matrix of 721 rows (t=721) and 102 columns (n=102). This corresponds to 721 observations made through time at 102 weather stations. `vswindnetD` is a `GNARnet` object containing `$edges` and `$dist`.
`edges` is a list of length 102, with `edges[[i]]` containing the vertices that node i is connected to.
`dist` is a list of length 102, with `dist[[i]]` containing the length of the vertices that node i is connected to. `vswindnet` is the same as `vswindnetD` except that all the distances are replaced by 1. `vswindnames` is a character vector of length 102 containing the wind speed site names and `vswindcoords` is a matrix with 102 rows (one for each wind station) and two columns providing the x and y coordinates of the weather stations.

Source

The base data were obtained from the http://wow.metoffice.gov.uk UK Met Office Weather-ObservationsWebsite distributed under the UK Open Government License https://www.nationalarchives.gov.uk/doc/open-government-licence/version/1/open-government-licence.htm Contains public sector information licensed under the Open Goverment Licence v1.0.

See Also

windnetplot

Examples

```
#
# The name entry for Bristol
#
vswindnames[77]
#[1] "BRIST"
#
# plot the distance network
#
## Not run: windnetplot()
```

wagner_plot *Produces a Wagner plot for the specified choice of covariates and/or time slices.*

Description

Produces a Wagner plot for comparing the netowrk autocorrelation and partial network autocorrerlation function values for a choice of maximum lag and maximum r-stage depth. The plot consists of rings linked to a time-slice or covariate, and indicates if the network autocorrelation differs for different time-slices and/or covariates. It is a visual for studying whether or not the network time series acts differently depending on lag or covariate. Starting from one and continuing to the outside, each ring corresponds to said choice of r-stage depth, the numbers on the outside ring are time-lags, and each dot corresponds to a specific time-slice or covariate.

Usage

```
wagner_plot(vts_frames, network_list, max_lag, max_stage, weight_matrices,
frame_names, same_net="no", viridis_color_option="viridis", size_option="absolute_val",
partial="no", wagner="yes")
```

Arguments

vts_frames	List containing the vector time series linked to each of the covariate and/or time-slice, which the Wagner plot compares.
network_list	List of network objects for which the Wagner plot compares network autocorrelation or partial network autocorrelation.
max_lag	Maximum lag for the Wagner plot.
max_stage	Maximum r-stage depth for the Wagner plot (i.e., the number of rings in the Wagner plot).
weight_matrices	List of weight matrices, each weight matrix corresponds to a particular choice of time-slice or covariate. If all the time-slices have the same weight matrix, then the argument is a list with said list.
frame_names	Indicates the name of each time-slice or covariate time series, order should be the same as in the weight matrices and vector time series lists.
same_net	Indicates whether or not all time-slices or covariates share the same weight matrix. Default choice is no, if the time-slices or covariates share the same weight matrix, then this argument should be set to "yes" (i.e., same_net = "yes").
viridis_color_option	Colour scale for the Corbit plot. The default option is `viridis`, each option is colout blind friendly; see `viridis` package.
size_option	Point size scale for the Corbit plot. The default is the absolute value of the network autocorrelation function (i.e., `|nacf(h, r)|` or `|pnacf(h, r)|`). Alternate option is the coefficient of determination coming from a global-α model with fixed lag and stage.

`partial`	Option for selecting between computing the network autocorrelation function or the partial network autocorrelation funciton. Default choice is network autocorrelation (i.e., partial="no"), change argument to "yes" for computing the partial network autocorrelation funciton.
`wagner`	Choice for distinguishing between Corbit and Wagner plots, default is set to Corbit (inner function call). For producing Wagner plots one should use `wagner_plot`.

Details

Wagner plots compare the network autocorrelation function (NACF) and partial network autocorrelation function (PNACF) values of different time-slices and/or covariate weights. Wagner plots are read in the same manner as Corbit plots `corbit_plot`, and include a legend on the right-hand side for distinguishing between covariates and/or time-slices. The point at the centre is the mean value of the NACF or PNACF values arising from the time-slices and/or covariate data splits. Essentially, if $c \in \{1, \ldots, C\}$, where $C \in \mathbb{N}$ is the number of covariates or time-slices, then the value at the centre is $\text{nacf}(h, r) = C^{-1} \sum_{c=1}^{C} \text{nacf}_c(h, r)$, where $\text{nacf}_c(h, r)$ is the NACF value corresponding to the covariate/time-slice c. The number of covariates and time-slices C must be equal to the length of the lists used for producing the Wagner plot.

Value

Produces the specified Wagner plot, does not return the network autocorrelaiton values.

Author(s)

Daniel Salnikov and Guy Nason.

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
## Not run:
#
# Produces a Wagner plot, which compares three stationary GNAR simulations,
# where the underlying network is fiveNet.
#
# Compute the weight matrix
W = weights_matrix(fiveNet)
#
# Simulate three stationary GNAR processe
sim1 <-  GNARsim(n = 100, net=fiveNet, alphaParams = list(c(0.1, 0.12, 0.16, 0.075, 0.21),
                                                          c(0.12, 0.14, 0.15, 0.6, 0.22)),
                 betaParams = list(c(0.1, 0.16), c(0.11, 0.14)))

sim2 <-  GNARsim(n = 100, net=fiveNet, alphaParams = list(rep(.25, 5)),
                 betaParams = list(c(0.1, 0.16)))

sim3 <- GNARsim(n = 100, net=fiveNet, alphaParams = list(rep(.25, 5), rep(0.13, 5)),
```

```
                betaParams = list(c(0.1, 0.16), c(0.11)))

# Produce NACF Wagner plot with the same network and weights matrix
wagner_plot(list(sim1, sim2, sim3), list(fiveNet), 10, 3, list(W),
  c("sim1", "sim2", "sim3"), same_net = "yes")
#
# Produce PNACF Wagner with different networks and weight matrices
wagner_plot(list(sim1, sim2, sim3), list(fiveNet, fiveNet, fiveNet), 10, 3, list(W, W, W),
  c("sim1", "sim2", "sim3"), same_net = "no", partial = "yes")

## End(Not run)
```

`weights_matrix` *Computes the weights matrix corresponding to the GNAR network object linked to the vector time series.*

Description

Computes the weights matrix with normalised weights (i.e., add up to one) for the network time series with underlying network provided by the user. If the network is unweighted, then each r-stage neighbour is considered equally relevant, i.e., $w_{ij} = \{\mathcal{N}_r(i)\}^{-1}\mathbb{I}(d(i,j) = r)$, where $\mathbb{I}$ is the indicator function and the distance is the shortest path in the underlying network.

Usage

```
weights_matrix(network, max_r_stage)
```

Arguments

`network`	Network linked to the vector time series under study, must be a GNARnet object.
`max_r_stage`	Longest shortest path for which weights are non-zero. If not specified, then its set equal to the upper bound, which is the longest shortest path in the underlying network.

Value

Weight matrix $\mathbf{W}$, each entry is the weight w_{ij} between a pair of nodes. The matrix is not symmetric, and each row adds up to one when considering r-stage neighbours for a particular r.

Author(s)

Daniel Salnikov and Guy Nason.

References

Nason, G.P., Salnikov, D. and Cortina-Borja, M. (2023) New tools for network time series with an application to COVID-19 hospitalisations. https://arxiv.org/abs/2312.00530

Examples

```
#
# Weights matrix linked to the mechanical ventilation beds time series.
# This network has a longest shortest path equal to six.
#
#data(fiveNet)
W_norm = weights_matrix(fiveNet, 6)
```

`windnetplot` *Produce bespoke plot of the wind data network*

Description

Plots the wind speed data network with distance information.

Usage

```
windnetplot()
```

Arguments

None.

Details

The wind speed data is to be found in the `vswind` data set. This function contains commands, using functionality from the `wordcloud` package, to plot the network, with node names and edges. Distances between nodes are plotted next to the edges.

See Also

`vswind`

Examples

```
## Not run: windplotnet()
```

Index