Package ‘tsnet’

July 22, 2025

Type Package

Title Fitting, Comparing, and Visualizing Networks Based on Time Series Data

Version 0.2.0

Maintainer Björn S. Siepe <`bjoernsiepe@gmail.com`>

Description Fit, compare, and visualize Bayesian graphical vector autoregressive (GVAR) network models using 'Stan'. These models are commonly used in psychology to represent temporal and contemporaneous relationships between multiple variables in intensive longitudinal data. Fitted models can be compared with a test based on matrix norm differences of posterior point estimates to quantify the differences between two estimated networks. See also Siepe, Kloft & Heck (2024) <doi:10.31234/osf.io/uwfjc>.

License GPL-3

Encoding UTF-8

LazyData true

URL `https://github.com/bsiepe/tsnet`

BugReports `https://github.com/bsiepe/tsnet/issues`

RoxygenNote 7.3.2

Imports cowplot, dplyr, ggdist, ggokabeito, ggplot2, loo, methods, posterior, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rlang, rstan (>= 2.18.1), rstantools (>= 2.3.1.1), stats, tidyr, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 4.1.0)

Biarch true

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

NeedsCompilation yes

Author Björn S. Siepe [aut, cre, cph] (ORCID:
<`https://orcid.org/0000-0002-9558-4648`>),
Matthias Kloft [aut] (ORCID: <`https://orcid.org/0000-0003-1845-6957`>),
Daniel W. Heck [ctb] (ORCID: <`https://orcid.org/0000-0002-6302-9252`>)

Repository CRAN

Date/Publication 2025-06-20 05:50:02 UTC

Contents

`tsnet-package` *The 'tsnet' package*

Description

Time Series Network Analysis with R

Author(s)

Maintainer: Björn S. Siepe <`bjoernsiepe@gmail.com`> (ORCID) [copyright holder]

Authors:

- Matthias Kloft <`kloft@uni-marburg.de`> (ORCID)

Other contributors:

- Daniel W. Heck <`daniel.heck@uni-marburg.de`> (ORCID) [contributor]

See Also

Useful links:

- https://github.com/bsiepe/tsnet
- Report bugs at https://github.com/bsiepe/tsnet/issues

`check_eigen` *Check Eigenvalues of Bayesian GVAR object*

Description

This function checks the eigenvalues of the Beta matrix (containing the temporal coefficients) to assure that the model is stationary. It uses the same check as the `graphicalVAR` package. The function calculates the eigenvalues of the Beta matrix and checks if the sum of the squares of the real and imaginary parts of the eigenvalues is less than 1. If it is, the VAR model is considered stable.

Usage

```
check_eigen(fitobj, verbose = TRUE)
```

Arguments

`fitobj`	A fitted Bayesian GVAR object. This can be a tsnet_fit object (obtained from `stan_gvar`, a `BGGM` object (obtained from `var_estimate`), or extracted posterior samples (obtained from `stan_fit_convert`.
`verbose`	Logical. If TRUE, a verbal summary of the results is printed. Default is `TRUE`.

Value

A list containing the eigenvalues and a verbal summary of the results.

Examples

```
data(fit_data)
fitobj <- fit_data[[1]]
result <- check_eigen(fitobj)
```

`compare_gvar` *Compare two Bayesian GVAR models*

Description

This function compares two Bayesian Graphical Vector Autoregressive models using matrix norms to test if the observed differences between two models is reliable. It computes the empirical distance between two models based on their point estimates and compares them using reference distributions created from their posterior distributions. Returns the p-value for the comparison based on a decision rule specified by the user. Details are available in Siepe, Kloft & Heck (2024) <doi:10.31234/osf.io/uwfjc>.

Usage

```
compare_gvar(
  fit_a,
  fit_b,
  cutoff = 5,
  dec_rule = "or",
  n_draws = 1000,
  comp = "frob",
  return_all = FALSE,
  sampling_method = "random",
  indices = NULL,
  burnin = 0
)
```

Arguments

`fit_a`	Fitted model object for Model A. This can be a tsnet_fit object (obtained from `stan_gvar`, a `BGGM` object (obtained from `var_estimate`, or extracted posterior samples (obtained from `stan_fit_convert`).
`fit_b`	Fitted model object for Model B. This can be a tsnet_fit object (obtained from `stan_gvar`, a `BGGM` object (obtained from `var_estimate`, or extracted posterior samples (obtained from `stan_fit_convert`).
`cutoff`	The percentage level of the test (default: `5`) as integer.
`dec_rule`	The decision rule to be used. Currently supports default `"or"` (comparing against two reference distributions) and "comb" (combining the reference distributions). The use of "or" is recommended, as "comb" is less stable.
`n_draws`	The number of draws to use for reference distributions (default: `1000`).
`comp`	The distance metric to use. Should be one of "frob" (Frobenius norm), "maxdiff" (maximum difference), or "l1" (L1 norm) (default: `"frob"`). The use of the Frobenius norm is recommended.
`return_all`	Logical indicating whether to return all distributions (default: `FALSE`). Has to be set to TRUE for plotting the results.

`sampling_method`	Draw sequential pairs of samples from the posterior, with certain distance between them ("sequential") or randomly from two halves of the posterior ("random"). The "random" method is preferred to account for potential autocorrelation between subsequent samples. Default: `"random"`.
`indices`	A list of "beta" and "pcor" indices specifying which elements of the matrices to consider when calculating distances. If `NULL` (default), all elements of both matrices are considered. If provided, only the elements at these indices are considered. If only one of the matrices should have indices, the other one should be NULL. This can be useful if you want to calculate distances based on a subset of the elements in the matrices.
`burnin`	The number of burn-in iterations to discard (default: `0`).

Value

A list (of class `compare_gvar`) containing the results of the comparison. The list includes:

`sig_beta`	Binary decision on whether there is a significant difference between the temporal networks of A and B
`sig_pcor`	Binary decision on whether there is a significant difference between the contemporaneous networks of A and B
`res_beta`	The null distribution for the temporal networks for both models
`res_pcor`	The null distribution for the contemporaneous networks for both models
`emp_beta`	The empirical distance between the two temporal networks
`emp_pcor`	The empirical distance between the two contemporaneous networks
`larger_beta`	The number of reference distances larger than the empirical distance for the temporal network
`larger_pcor`	The number of reference distances larger than the empirical distance for the temporal network
`arguments`	The arguments used in the function call

Examples

```
# use internal fit data of two individuals
data(fit_data)
test_res <- compare_gvar(fit_data[[1]],
fit_data[[2]],
n_draws = 100,
return_all = TRUE)
print(test_res)
```

`fit_data` *Example Posterior Samples*

Description

This dataset contains posterior samples of beta coefficients and partial correlations for two individuals. It was generated by fitting a GVAR model using `stan_gvar` with three variables from the `ts_data` dataset.

Usage

```
data(fit_data)
```

Format

‘fit_data‘ A list with two elements, each containing posterior samples for one individual.

Details

The list contains two elements, each containing posterior samples for one individual. The samples were extracted using the `stan_fit_convert` function. For each individual, the list elements contain the posterior means of the beta coefficients ("beta_mu") and the posterior means of the partial correlations ("pcor_mu"). The "fit" element contains all 1000 posterior samples of the beta coefficients and partial correlations.

Source

The data is generated using the `stan_gvar` function on subsets of the `ts_data` time series data.

`get_centrality` *Compute Centrality Measures*

Description

This function computes various network centrality measures for a given GVAR fit object. Centrality measures describe the "connectedness" of a variable in a network, while density describes the networks’ overall connectedness. Specifically, it computes the in-strength, out-strength, contemporaneous strength, temporal network density, and contemporaneous network density. The result can then be visualized using `plot_centrality`.

Usage

```
get_centrality(fitobj, burnin = 0, remove_ar = TRUE)
```

Arguments

`fitobj`	Fitted model object for a Bayesian GVAR model. This can be 'tsnet_fit' object (obtained from `stan_gvar`, a BGGM object (obtained from `var_estimate` in `BGGM`), or extracted posterior samples (obtained from `stan_fit_convert`).
`burnin`	An integer specifying the number of initial samples to discard as burn-in. Default is `0`.
`remove_ar`	A logical value specifying whether to remove the autoregressive effects for centrality calculation. Default is `TRUE`. This is only relevant for the calculation of temporal centrality/density measures.

Value

A list containing the following centrality measures:

- `instrength`: In-strength centrality.
- `outstrength`: Out-strength centrality.
- `strength`: Contemporaneous strength centrality.
- `density_beta`: Temporal network density.
- `density_pcor`: Contemporaneous network density.

Examples

```
# Use first individual from example fit data from tsnet
data(fit_data)
centrality_measures <- get_centrality(fit_data[[1]])
```

`plot.compare_gvar` *Plot compare_gvar*

Description

This function is a plotting method for the class produced by `compare_gvar`. It generates a plot showing the density of posterior uncertainty distributions for distances and the empirical distance value for two GVAR models.

Usage

```
## S3 method for class 'compare_gvar'
plot(x, name_a = NULL, name_b = NULL, ...)
```

Arguments

`x`	An object of class `compare_gvar`.
`name_a`	Optional. The name for model A. If provided, it replaces "mod_a" in the plot.
`name_b`	Optional. The name for model B. If provided, it replaces "mod_b" in the plot.
`...`	Additional arguments to be passed to the plotting functions.

Details

The function first checks if the full reference distributions of `compare_gvar` are saved using the argument `return_all` set to `TRUE`. If not, an error is thrown.

Using the "name_a" and "name_b" arguments allows for custom labeling of the two models in the plot.

The function generates two density plots using `ggplot2`, one for the temporal network (beta) and another for the contemporaneous network (pcor). The density distributions are filled with different colors based on the corresponding models (mod_a and mod_b). The empirical distances between the networks are indicated by red vertical lines.

Value

A `ggplot` object representing the density plots of the posterior uncertainty distributions for distances and the empirical distance for two GVAR models.

Examples

```
data(fit_data)
test_res <- compare_gvar(fit_data[[1]],
fit_data[[2]],
n_draws = 100,
return_all = TRUE)
plot(test_res)
```

`plot_centrality` *Plot Centrality Measures*

Description

This function creates a plot of various centrality measures for a given object. The plot can be either a "tiefighter" plot or a "density" plot. The "tiefighter" plot shows the centrality measures for each variable with uncertainty bands, while the "density" plot shows the full density of the centrality measures.

Usage

```
plot_centrality(obj, plot_type = "tiefighter", cis = 0.95)
```

Arguments

`obj`	An object containing the centrality measures obtained from `get_centrality`.
`plot_type`	A character string specifying the type of plot. Accepts "tiefighter" or "density". Default is `"tiefighter"`.
`cis`	A numeric value specifying the credible interval. Must be between 0 and 1 (exclusive). Default is `0.95`.

Value

A ggplot object visualizing the centrality measures. For a "tiefighter" plot, each point represents the mean centrality measure for a variable, and the bars represent the credible interval. In a "density" plot, distribution of the centrality measures is visualized.

Examples

```
data(fit_data)
obj <- get_centrality(fit_data[[1]])
  plot_centrality(obj,
  plot_type = "tiefighter",
  cis = 0.95)
```

`posterior_plot` *posterior_plot*

Description

Plots posterior distributions of the parameters of the temporal or the contemporaneous networks of a GVAR model. The posterior distributions are visualized as densities in a matrix layout.

Usage

```
posterior_plot(fitobj, mat = "beta", cis = c(0.8, 0.9, 0.95))
```

Arguments

`fitobj`	Fitted model object. This can be a tsnet_fit object (obtained from `stan_gvar`, a `BGGM` object (obtained from `var_estimate`.
`mat`	A matrix to use for plotting. Possibilities include "beta" (temporal network) and "pcor" (contemporaneous network). Default is `"beta"` (temporal network).
`cis`	A numeric vector of credible intervals to use for plotting. Default is `c(0.8, 0.9, 0.95)`.

Details

In the returned plot, posterior distributions for every parameter are shown. Lagged variables are displayed along the vertical line of the grid, and non-lagged variables along the horizontal line of the grids.

Value

A `ggplot` object representing the posterior distributions of the parameters of the temporal or the contemporaneous networks of a GVAR model.

Examples

```
# Load simulated time series data
data(ts_data)
example_data <- ts_data[1:100,1:4]

# Estimate a GVAR model
fit <- stan_gvar(example_data, n_chains = 2)

# Extract posterior samples
posterior_plot(fit)
```

`post_distance_within` *Calculates distances between pairs of posterior samples using the posterior samples or posterior predictive draws*

Description

This function computes distances between posterior samples of a single fitted GVAR model. Thereby, it calculates the uncertainty contained in the posterior distribution, which can be used as a reference to compare two modes. Distances can be obtained either from posterior samples or posterior predictive draws. The distance between two models can currently be calculated based on three options: Frobenius norm, maximum difference, or L1 norm. Used within `compare_gvar`. The function is not intended to be used directly by the user.

Usage

```
post_distance_within(
  fitobj,
  comp,
  pred,
  n_draws = 1000,
  sampling_method = "random",
  indices = NULL,
  burnin = 0
)
```

Arguments

`fitobj`	Fitted model object. This can be a tsnet_fit object (obtained from `stan_gvar`, a `BGGM` object (obtained from `var_estimate`, or extracted posterior samples (obtained from `stan_fit_convert`).
`comp`	The distance metric to use. Should be one of "frob" (Frobenius norm), "maxdiff" (maximum difference), or "l1" (L1 norm) (default: `"frob"`). The use of the Frobenius norm is recommended.
`pred`	A logical indicating whether the input is posterior predictive draws (TRUE) or posterior samples (FALSE). Default: `FALSE`

`n_draws`	The number of draws to use for reference distributions (default: `1000`).
`sampling_method`	Draw sequential pairs of samples from the posterior, with certain distance between them ("sequential") or randomly from two halves of the posterior ("random"). The "random" method is preferred to account for potential autocorrelation between subsequent samples. Default: `"random"`.
`indices`	A list of "beta" and "pcor" indices specifying which elements of the matrices to consider when calculating distances. If `NULL` (default), all elements of both matrices are considered. If provided, only the elements at these indices are considered. If only one of the matrices should have indices, the other one should be NULL. This can be useful if you want to calculate distances based on a subset of the elements in the matrices.
`burnin`	The number of burn-in iterations to discard (default: `0`).

Value

A list of distances between the specified pairs of fitted models. The list has length equal to the specified number of random pairs. Each list element contains two distance values, one for beta coefficients and one for partial correlations.

Examples

```
data(fit_data)
post_distance_within(fitobj = fit_data[[1]],
comp = "frob",
pred = FALSE,
n_draws = 100)
```

`print.compare_gvar` *Print method for compare_gvar objects*

Description

This function prints a summary of the Norm-Based Comparison Test for a `compare_gvar` object.

Usage

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

Arguments

`x`	A test object obtained from `compare_gvar`
`...`	Additional arguments to be passed to the print method. (currently not used)

Details

This function prints a summary of the Norm-Based Comparison Test for a `compare_gvar` object. in the temporal and contemporaneous networks, as well as the number of reference distances that were larger than the empirical distance for each network.

Value

Prints a summary of the Norm-Based Comparison Test to the console

Examples

```
# Load example fits
data(fit_data)

# Perform test
test_res <- compare_gvar(fit_data[[1]], fit_data[[2]], n_draws = 100)

# Print results
print(test_res)
```

`print.tsnet_fit` *Print method for tsnet_fit objects*

Description

This method provides a summary of the Bayesian GVAR model fitted with `stan_gvar`. It prints general information about the model, including the estimation method and the number of chains and iterations It also prints the posterior mean of the temporal and contemporaneous coefficients.

Usage

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

Arguments

`x`	A tsnet_fit object.
`...`	Additional arguments passed to the print method (currently not used).

Value

Prints a summary to the console.

Examples

```
# Load example data
data(ts_data)
example_data <- ts_data[1:100,1:3]

# Fit the model
fit <- stan_gvar(example_data,
                 method = "sampling",
                 cov_prior = "IW",
                 n_chains = 2)

print(fit)
```

`stan_fit_convert` *Convert Stan Fit to Array of Samples*

Description

This function converts a Stan fit object into an array of samples for the temporal coefficients and the innovation covariance or partial correlation matrices. It supports `rstan` as a backend. It can be used to convert models fit using `stan_gvar` into 3D arrays, which is the standard data structure used in `tsnet`. The function allows to select which parameters should be returned.

Usage

```
stan_fit_convert(stan_fit, return_params = c("beta", "sigma", "pcor"))
```

Arguments

`stan_fit`	A Stan fit object obtained from rstan or a tsnet_fit object from `stan_gvar`.
`return_params`	A character vector specifying which parameters to return. Options are "beta" (temporal network), "sigma" (innovation covariance), and "pcor" (partial correlations). Default is `c("beta","sigma", "pcor")`.

Value

A list containing 3D arrays for the selected parameters. Each array represents the posterior samples for a parameter, and each slice of the array represents a single iteration.

Examples

```
data(ts_data)
example_data <- ts_data[1:100,1:3]
fit <- stan_gvar(data = example_data,
                 n_chains = 2,
                 n_cores = 1)
samples <- stan_fit_convert(fit, return_params = c("beta", "pcor"))
```

`stan_gvar` *Fit Bayesian Graphical Vector Autoregressive (GVAR) Models with Stan*

Description

This function fits a Bayesian GVAR model to the provided data using Stan. The estimation procedure is described further in Siepe, Kloft & Heck (2023) <doi:10.31234/osf.io/uwfjc>. The current implementation allows for a normal prior on the temporal effects and either an Inverse Wishart or an LKJ prior on the contemporaneous effects. `rstan` is used as a backend for fitting the model in Stan. Data should be provided in long format, where the columns represent the variables and the rows represent the time points. Data are automatically z-scaled for estimation. The model currently does not support missing data.

Usage

```
stan_gvar(
  data,
  beep = NULL,
  priors = NULL,
  method = "sampling",
  cov_prior = "IW",
  rmv_overnight = FALSE,
  iter_sampling = 500,
  iter_warmup = 500,
  n_chains = 4,
  n_cores = 1,
  center_only = FALSE,
  return_all = TRUE,
  ahead = 0,
  ...
)
```

Arguments

`data`	A data frame or matrix containing the time series data of a single subject. The data should be in long format, where the columns represent the variables and the rows represent the time points. See the example data `ts_data` for the correct format.
`beep`	A vector of beeps with length of `nrow(data)`. The beep indicator can be used to remove overnight effects from the last beep of a day to the first beep of the next day. This should be a vector of positive integers. If left empty, the function will assume that there are no overnight effects to remove.
`priors`	A list of prior distributions for the model parameters. This should be a named list, with names corresponding to the parameter names and values corresponding to the prior distributions. The following priors can be specified:

	• `prior_Beta_loc` A matrix of the same dimensions as the beta matrix ‘B‘ containing the mean of the prior distribution for the beta coefficients. • `prior_Beta_scale` A matrix of the same dimensions as the beta matrix ‘B‘ containing the standard deviation of the prior distribution for the beta coefficients.
`method`	A string indicating the method to use for fitting the model. Options are "sampling" (for MCMC estimation) or "variational" (for variational inference). We currently recommend only using MCMC estimation. Default: `"sampling"`.
`cov_prior`	A string indicating the prior distribution to use for the covariance matrix. Options are "LKJ" or "IW" (Inverse-Wishart). Default: `"LKJ"`.
`rmv_overnight`	A logical indicating whether to remove overnight effects. Default is `FALSE`. If ‘TRUE‘, the function will remove overnight effects from the last beep of a day to the first beep of the next day. This requires the beep argument to be specified.
`iter_sampling`	An integer specifying the number of iterations for the sampling method. Default is `500`.
`iter_warmup`	An integer specifying the number of warmup iterations for the sampling method. Default is `500`.
`n_chains`	An integer specifying the number of chains for the sampling method. Default is 4. If variational inference is used, the number of iterations is calculated as `iter_sampling*n_chains`.
`n_cores`	An integer specifying the number of cores to use for parallel computation. Default is `1`. `rstan` is used for parallel computation.
`center_only`	A logical indicating whether to only center (and not scale) the data. Default is `FALSE`.
`return_all`	A logical indicating whether to return all model inputs, including the data and prior objects. Default is `TRUE`.
`ahead`	An integer specifying the forecast horizon. Default is `0`. If ‘ahead‘ is greater than 0, the function will return posterior predictive forecasts for the specified number of time points. This functionality is experimental and may not work as expected.
`...`	Additional arguments passed to the `rstan::sampling` or `rstan::vb` function.

Details

General Information

In a Graphical Vector Autoregressive (GVAR) model of lag 1, each variable is regressed on itself and all other variables at the previous timepoint to obtain estimates of the temporal association between variables (encapsulated in the beta matrix). This is the "Vector Autoregressive" part of the model. Additionally, the innovation structure at each time point (which resembles the residuals) is modeled to obtain estimates of the contemporaneous associations between all variables (controlling for the lagged effects). This is typically represented in the partial correlation (pcor) matrix. If the model is represented and interpreted as a network, variables are called *nodes*, *edges* represent the statistical association between the nodes, and *edge weights* quantify the strength of these associations.

Model

Let Y be a matrix with n rows and p columns, where n_t is the number of time points and p is the number of variables. The GVAR model is given by the following equations:

$$Y_t = B * Y_{t-1} + \zeta_t$$

$$\zeta_t \sim N(0, \Sigma)$$

where B is a $p \times p$ matrix of VAR coefficients between variables i and j (β_{ij}), ζ_t contains the innovations at time point t, and Σ is a $p \times p$ covariance matrix. The inverse of Σ is the precision matrix, which is used to obtain the partial correlations between variables (ρ_{ij}). The model setup is explained in more detail in Siepe, Kloft & Heck (2023) <doi:10.31234/osf.io/uwfjc>.

Prior Setup

For the $p \times p$ temporal matrix B (containing the β coefficients), we use a normal prior distribution on each individual parameter:

$$\beta_{ij} \sim N(PriorBetaLoc_{ij}, PriorBetaScale_{ij})$$

where `PriorBetaLoc` is the mean of the prior distribution and `PriorBetaScale` is the standard deviation of the prior distribution. The default prior is a weakly informative normal distribution with mean 0 and standard deviation 0.5. The user can specify a different prior distribution by a matrix `prior_Beta_loc` and a matrix `prior_Beta_scale` with the same dimensions as B.

Both a Lewandowski-Kurowicka-Joe (LKJ) and an Inverse-Wishart (IW) distribution can be used as a prior for the contemporaneous network. However, the LKJ prior does not allow for direct specifications of priors on the partial correlations. We implemented a workaround to enable priors on specific partial correlations (described below). We consider this feature experimental would advise users wishing to implement edge-specific priors in the contemporaneous network to preferentially use IW priors.

The LKJ prior is a distribution on the correlation matrix, which is parameterized by the shape parameter η. To enable edge-specific priors on the partial correlations, we use the workaround of a "joint" prior that, in addition to the LKJ on the correlation matrix itself, allows for an additional beta prior on each of the partial correlations. We first assigned an uninformed LKJ prior to the Cholesky factor decomposition of the correlation matrix of innovations:

$$\Omega_L \sim LKJ - Cholesky(\eta)$$

For $\eta = 1$, this implies a symmetric marginal scaled beta distribution on the zero-order correlations ω_{ij}.

$$(\omega_{ij} + 1)/2 \sim Beta(p/2, p/2)$$

We can then obtain the covariance matrix and, subsequently, the precision matrix (see Siepe, Kloft & Heck, 2023, for details). The second part of the prior is a beta prior on each partial correlation ρ_{ij} (obtained from the off-diagonal elements of the precision matrix). This prior was assigned by transforming the partial correlations to the interval of $[0, 1]$ and then assigning a proportional (mean-variance parameterized) beta prior:

$$(\rho_{ij} + 1)/2 \sim Beta_{prop}(PriorRhoLoc, PriorRhoScale)$$

A beta location parameter of 0.5 translates to an expected correlation of 0. The variance parameter of $\sqrt{(}0.5)$ implies a uniform distribution of partial correlations. The user can specify a different prior distribution by a matrix `prior_Rho_loc` and a matrix `prior_Rho_scale` with the same dimensions as the partial correlation matrix. Additionally, the user can change eta via the `prior_Eta` parameter.

The Inverse-Wishart prior is a distribution on the innovation covariance matrix Σ:

$$\Sigma \sim IW(\nu, S)$$

where ν is the degrees of freedom and S is the scale matrix. We here use the default prior of

$$nu = delta + p - 1$$

for the degrees of freedom, where δ is defined as $s_\rho^{-1} - 1$ and s_ρ is the standard deviation of the implied marginal beta distribution of the partial correlations. For the scale matrix S, we use the identity matrix I_p of order p. The user can set a prior on the expected standard deviation of the partial correlations by specifying a `prior_Rho_marginal` parameter. The default value is 0.25, which has worked well in a simulation study. Additionally, the user can specify a `prior_S` parameter to set a different scale matrix.

Sampling The model can be fitted using either MCMC sampling or variational inference via `rstan`. Per default, the model is fitted using the Stan Hamiltonian Monte Carlo (HMC) No U-Turn (NUTS) sampler with 4 chains, 500 warmup iterations and 500 sampling iterations. We use a default target average acceptance probability `adapt_delta` of 0.8. As the output is returned as a standard `stanfit` object, the user can use the `rstan` package to extract and analyze the results and obtain convergence diagnostics.

Value

A `tsnet_fit` object in list format. The object contains the following elements:

`fit`	A stanfit object containing the fitted model.
`arguments`	The number of variables "p", the number of time points "n_t", the column names "cnames", and the arguments used in the function call.

Examples

```
# Load example data
data(ts_data)
example_data <- ts_data[1:100,1:3]

# Fit the model
fit <- stan_gvar(example_data,
                 method = "sampling",
                 cov_prior = "IW",
                 n_chains = 2)
print(fit)
```

`ts_data` *Simulated Time Series Dataset*

Description

This dataset contains a simulated time series dataset for two individuals generated using the `graphicalVAR` package. The dataset is useful for testing and demonstrating the functionality of the package.

Usage

```
data(ts_data)
```

Format

‘ts_data‘ A data frame with 500 rows and 7 columns.

id A character string identifier for the individual. There are two unique ids, representing two individuals.

V1-V6 These columns represent six different variables in the time series data.

Details

The dataset consists of 250 observations each of 6 variables for two individuals. The variables V1-V6 represent simulated time series data generated using the graphicalVARsim function from the graphicalVAR package. The ’id’ column contains a character string as identifier of the two individuals. The data have been standardized to have zero mean and unit variance.

Source

Simulated using the `graphicalVARsim` function in the `graphicalVAR` package.

Index