Package ‘statespacer’

November 19, 2020

Version 0.4.0
Date 2020-11-19
Title State Space Modelling in 'R'

URL https://DylanB95.github.io/statespacer/,
https://github.com/DylanB95/statespacer/

BugReports https://github.com/DylanB95/statespacer/issues/
License MIT + file LICENSE
RoxygenNote 7.1.1
RdMacros Rdpack
Depends R (>= 3.6)
Imports Rdpack, stats, Rcpp
Suggests datasets, graphics, knitr, numDeriv (>= 2016.8-1.1), optimx (>= 2020-4.2), rmarkdown, YieldCurve (>= 4.1)
VignetteBuilder knitr
LinkingTo Rcpp, RcppArmadillo
NeedsCompilation yes
Author Dylan Beijers [aut, cre]
Maintainer Dylan Beijers <dylanbeijers@gmail.com>
Repository CRAN
Date/Publication 2020-11-19 19:20:32 UTC
BlockMatrix

Combine Matrices into a Block Diagonal Matrix

Description

Creates a block diagonal matrix with its arguments as the blocks.

Usage

BlockMatrix(...)

Arguments

... Matrices that should be put on the diagonal.

Details

BlockMatrix() tries to coerce its arguments to a matrix, using `as.matrix`.

Value

Block diagonal matrix having the specified matrices on its diagonal.

Author(s)

Dylan Beijers, <dylanbeijers@gmail.com>

Examples

BlockMatrix(diag(ceiling(9 * stats::rnorm(5))), matrix(1:8, 4, 2), c(14, 8))
Cholesky

Construct a Valid Variance - Covariance Matrix

Description

Constructs a valid variance - covariance matrix by using the Cholesky LDL decomposition.

Usage

Cholesky(param = NULL, format = NULL, decompositions = TRUE)

Arguments

- **param**: Vector containing the parameters used to construct the variance - covariance matrix.
- **format**: Matrix representing the format for the Loading matrix L and Diagonal matrix D. The lower triangular part of the format is used as the format for the Loading matrix L. The diagonal of the format is used as the format for the Diagonal matrix D. Must be a matrix.
- **decompositions**: Boolean indicating whether the loading and diagonal matrix of the Cholesky decomposition, and the correlation matrix and standard deviations should be returned.

Details

The elements of `param` are then distributed along the non-zero elements of the loading and diagonal matrix. The parameters for the diagonal matrix are transformed using \( \exp(2 \times x) \).

Value

A valid variance - covariance matrix. If decompositions = TRUE then it returns a list containing:

- **cov_mat**: The variance - covariance matrix.
- **loading_matrix**: The loading matrix of the Cholesky decomposition.
- **diagonal_matrix**: The diagonal matrix of the Cholesky decomposition.
- **correlation_matrix**: Matrix containing the correlations.
- **stdev_matrix**: Matrix containing the standard deviations on the diagonal.

Author(s)

Dylan Beijers, <dylanbeijers@gmail.com>

Examples

```r
format <- diag(1, 2, 2)
format[2, 1] <- 1
Cholesky(param = c(2, 4, 1), format = format, decompositions = TRUE)
```
CoeffARMA

Transform arbitrary matrices into ARMA coefficient matrices

Description

Creates coefficient matrices for which the characteristic polynomial corresponds to a stationary process. See Ansley and Kohn (1986) for details about the transformation used.

Usage

CoeffARMA(A, variance = NULL, ar = 1, ma = 0)

Arguments

A
An array of arbitrary square matrices in the multivariate case, or a vector of arbitrary numbers in the univariate case.

variance
A variance-covariance matrix. Note: variance not needed for the univariate case!

ar
The order of the AR part.

ma
The order of the MA part.

Value

If multivariate, a list containing:

• An array of coefficient matrices for the AR part.
• An array of coefficient matrices for the MA part.

If univariate, a list containing:

• A vector of coefficients for the AR part.
• A vector of coefficients for the MA part.

Author(s)

Dylan Beijers, <dylanbeijers@gmail.com>

References


Examples

CoeffARMA(A = stats::rnorm(2), ar = 1, ma = 1)
predict.statespacer      State Space Model Forecasting

Description

Produces forecasts using a fitted State Space Model.

Usage

```r
## S3 method for class 'statespacer'
predict(
  object,
  addvar_list_fc = NULL,
  level_addvar_list_fc = NULL,
  self_spec_list_fc = NULL,
  forecast_period = 1,
  ...
)
```

Arguments

- `object` A statespacer object as returned by `statespacer`.
- `addvar_list_fc` A list containing the explanatory variables for each of the dependent variables. The list should contain `p` (number of dependent variables) elements. Each element of the list should be a `forecast_period` x `k_p` matrix, with `k_p` being the number of explanatory variables for the `p`th dependent variable. If no explanatory variables should be added for one of the dependent variables, then set the corresponding element to `NULL`.
- `level_addvar_list_fc` A list containing the explanatory variables for each of the dependent variables. The list should contain `p` (number of dependent variables) elements. Each element of the list should be a `forecast_period` x `k_p` matrix, with `k_p` being the number of explanatory variables for the `p`th dependent variable. If no explanatory variables should be added for one of the dependent variables, then set the corresponding element to `NULL`.
- `self_spec_list_fc` A list containing the specification of the self specified component. Does not have to be specified if it does not differ from `self_spec_list` as passed on to `statespacer`. If some system matrices are time-varying then you should specify this argument. See `statespacer` for details about the format that must be followed for this argument.
- `forecast_period` Number of time steps to forecast ahead.
- `...` Arguments passed on to the predict generic. Should not be used!
Value

A list containing the forecasts and corresponding uncertainties. In addition, it returns the com-ponents of the forecasts, as specified by the State Space model.

Author(s)

Dylan Beijers, <dylanbeijers@gmail.com>

References


Examples

```r
# Fits a local level model for the Nile data
dimension(datasets) <- matrix(Nile)
fit <- statespacer(initial = 10, y = y, local_level.ind = TRUE)

# Obtain forecasts for 10 steps ahead using the fitted model
fc <- predict(fit, forecast_period = 10)

# Plot the forecasts
plot(1:10, fc$y_fc, type = "l")
```

---

**SimSmoother**

**Generating Random Samples using the Simulation Smoother**

Description

Draws random samples of the specified model conditional on the observed data.

Usage

`SimSmoother(object, nsim = 1, components = TRUE)`

Arguments

- `object` A statespacer object as returned by `statespacer`.
- `nsim` Number of random samples to draw. Defaults to 1.
- `components` Boolean indicating whether the components of the model should be extracted in each of the random samples.
Value

A list containing the simulated state parameters and disturbances. In addition, it returns the components as specified by the State Space model if components = TRUE. Each of the objects are arrays, where the first dimension equals the number of time points, the second dimension the number of state parameters, disturbances, or dependent variables, and the third dimension equals the number of random samples nsim.

Author(s)

Dylan Beijers, <dylanbeijers@gmail.com>

Examples

# Fits a local level model for the Nile data
library(datasets)
y <- matrix(Nile)
fit <- statespacer(initial = 10, y = y, local_level_ind = TRUE)

# Obtain random sample using the fitted model
sim <- SimSmoother(fit, nsim = 1, components = TRUE)

# Plot the simulated level against the smoothed level of the original data
plot(sim$level[, 1, 1], type = 'p')
lines(fit$smoothed$level, type = 'l')

statespacer  State Space Model Fitting

Description

Fits a State Space model as specified by the user.

Usage

statespacer(
  y,
  H_format = NULL,
  local_level_ind = FALSE,
  slope_ind = FALSE,
  BSM_vec = NULL,
  cycle_ind = FALSE,
  addvar_list = NULL,
  level_addvar_list = NULL,
  arima_list = NULL,
  sarima_list = NULL,
  self_spec_list = NULL,
  exclude_level = NULL,
  exclude_slope = NULL,
exclude_BSM_list = lapply(BSM_vec, function(x) 0),
exclude_cycle_list = list(0),
exclude_arima_list = lapply(arima_list, function(x) 0),
deamping_factor_ind = rep(TRUE, length(exclude_cycle_list)),
format_level = NULL,
format_slope = NULL,
format_BSM_list = lapply(BSM_vec, function(x) NULL),
format_cycle_list = lapply(exclude_cycle_list, function(x) NULL),
format_addvar = NULL,
format_level_addvar = NULL,
fit = TRUE,
inital = 0,
method = "BFGS",
control = list(),
collapse = FALSE,
diagnostics = TRUE,
standard_errors = NULL,
verbose = FALSE
)

Arguments

y N x p matrix containing the N observations of the p dependent variables.
H_format Format of the H system matrix, the variance - covariance matrix of the observation equation.
local_level_ind Boolean indicating whether a local level should be added to the state space model.
slope_ind Boolean indicating whether a local level + slope should be added to the state space model.
BSM_vec Vector containing the BSM seasonalities that have to be added to the state space model.
cycle_ind Boolean indicating whether a cycle has to be added to the state space model.
addvar_list A list containing the explanatory variables for each of the dependent variables. The list should contain p (number of dependent variables) elements. Each element of the list should be a N x k_p matrix, with k_p being the number of explanatory variables for the pth dependent variable. If no explanatory variables should be added for one of the dependent variables, then set the corresponding element to NULL.
level_addvar_list A list containing the explanatory variables for each of the dependent variables. The list should contain p (number of dependent variables) elements. Each element of the list should be a N x k_p matrix, with k_p being the number of explanatory variables for the pth dependent variable. If no explanatory variables should be added for one of the dependent variables, then set the corresponding element to NULL.
arima_list Specifications of the ARIMA components, should be a list containing vectors of length 3 with the following format: c(AR, I, MA). Should be a list to allow different ARIMA models for different sets of dependent variables. Note: The AR and MA coefficients are constrained such that the AR component is stationary, and the MA component is invertible. See Ansley and Kohn (1986) for details about the transformation used.

sarima_list Specifications of the SARIMA components, should be a list containing lists that contain 4 named vectors. Vectors should be named: "s", "ar", "i", "ma". Should be a list of lists to allow different SARIMA models for different sets of dependent variables. Note: The AR and MA coefficients are constrained such that the AR components are stationary, and the MA components are invertible. See Ansley and Kohn (1986) for details about the transformation used. Note: For multivariate models, the order of "s" matters, as matrix multiplication is not commutative!

self_spec_list A list containing the specification of the self specified component. See the Details section for extensive details about the format that must be followed for this argument.

exclude_level Vector containing the dependent variables that should not get a local level.

exclude_slope Vector containing the dependent variables that should not get a slope.

exclude_BSM_list List of vectors, each vector containing the dependent variables that should not get the corresponding BSM component.

exclude_cycle_list The dependent variables that should not get the corresponding cycle component. Should be a list of vectors to allow different dependent variables to be excluded for different cycles.

exclude_arima_list The dependent variables that should not be involved in the corresponding ARIMA component. Should be a list of vectors to allow different dependent variables to be excluded for different ARIMA components.

exclude_sarima_list The dependent variables that should not be involved in the corresponding SARIMA component. Should be a list of vectors to allow different dependent variables to be excluded for different SARIMA components.

damping_factor_ind Boolean indicating whether a damping factor should be included. Must be a vector if multiple cycles are included, to indicate which cycles should include a damping factor.

format_level Format of the Q_level system matrix the variance - covariance matrix of the level state equation.

format_slope Format of the Q_slope system matrix, the variance - covariance matrix of the slope state equation.

format_BSM_list Format of the Q_BSM system matrix, the variance - covariance matrix of the BSM state equation. Should be a list to allow different formats for different seasonality periods.
format_cycle_list
  Format of the Q_cycle system matrix, the variance - covariance matrix of the
cycle state equation. Should be a list to allow different formats for different
cycles.

format_addvar
  Format of the Q_addvar system matrix, the variance - covariance matrix of the
explanatory variables state equation.

format_level_addvar
  Format of the Q_level_addvar system matrix, the variance - covariance matrix
of the explanatory variables of the level state equation.

fit
  Boolean indicating whether the model should be fit by an iterative optimisation
procedure. If FALSE, the model is only evaluated at the initial values.

initial
  Vector of initial values for the parameter search. The initial values are recycled
or truncated if too few or too many values have been specified.

method
  Method that should be used by the optim or optimr function to estimate the
parameters. Only used if fit = TRUE.

collapse
  Boolean indicating whether the observation vector should be collapsed. Should
only be set to TRUE if the dimensionality of the observation vector exceeds the
dimensionality of the state vector. If this is the case, computational gains can be
achieved by collapsing the observation vector.

diagnostics
  Boolean indicating whether diagnostical tests should be computed. Defaults to
TRUE.

standard_errors
  Boolean indicating whether standard errors should be computed. numDeriv
must be installed in order to compute the standard errors! Defaults to TRUE if
numDeriv is available.

verbose
  Boolean indicating whether the progress of the optimisation procedure should
be printed. Only used if fit = TRUE.

Details
To fit the specified State Space model, one occasionally has to pay careful attention to the initial
values supplied. See vignette("dictionary","statespacer") for details. Initial values should
not be too large, as some parameters use the transformation exp(2x) to ensure non-negative values,
they should also not be too small as some variances might become relatively too close to 0, relative
to the magnitude of y.

If a component is specified without a format, then the format defaults to a diagonal format.

self_spec_list provides a means to incorporate a self-specified component into the State Space
model. This argument can only contain any of the following items, of which some are mandatory:

• H_spec: Boolean indicating whether the H matrix is self-specified. Should be TRUE, if you
  want to specify the H matrix yourself.

• state_num (mandatory): The number of state parameters introduced by the self-specified
  component. Must be 0 if only H is self-specified.
• **param_num**: The number of parameters needed by the self-specified component. Must be specified and greater than 0 if parameters are needed.

• **sys_mat_fun**: A function returning a list of system matrices that are constructed using the parameters. Must have `param` as an argument. The items in the list returned should have any of the following names: `Z`, `Tmat`, `R`, `Q`, `a1`, `P_star`, `H`. Note: Only the system matrices that depend on the parameters should be returned by the function!

• **sys_mat_input**: A list containing additional arguments to `sys_mat_fun`.

• **Z**: The `Z` system matrix if it does not depend on the parameters.

• **Tmat**: The `T` system matrix if it does not depend on the parameters.

• **R**: The `R` system matrix if it does not depend on the parameters.

• **Q**: The `Q` system matrix if it does not depend on the parameters.

• **a1**: The initial guess of the state vector. Must be a matrix with one column.

• **P_inf**: The initial diffuse part of the variance - covariance matrix of the initial state vector. Must be a matrix.

• **P_star**: The initial non-diffuse part of the variance - covariance matrix of the initial state vector if it does not depend on the parameters. Must be a matrix.

• **H**: The `H` system matrix if it does not depend on the parameters.

• **transform_fun**: Function that returns transformed parameters for which standard errors have to be computed. Must have `param` as an argument.

• **transform_input**: A list containing additional arguments to `transform_fun`.

• **state_only**: The indices of the self specified state that do not play a role in the observation equations, but only in the state equations. Should only be used if you want to use `collapse = TRUE` and have some state parameters that do not play a role in the observation equations. Does not have to be specified for `collapse = FALSE`.

Note: System matrices should only be specified once and need to be specified once! That is, system matrices that are returned by `sys_mat_fun` should not be specified directly, and vice versa. So, system matrices need to be either specified directly, or be returned by `sys_mat_fun`. An exception holds for the case where you **only** want to specify `H` yourself. This will not be checked, so be aware of erroneous output if you do not follow the guidelines of specifying `self_spec_list`. If time-varying system matrices are required, return an array for the time-varying system matrix instead of a matrix.

**Value**

A statespacer object containing:

• **function_call**: A list containing the input to the function.

• **system_matrices**: A list containing the system matrices of the State Space model.

• **predicted**: A list containing the predicted components of the State Space model.

• **filtered**: A list containing the filtered components of the State Space model.

• **smoothed**: A list containing the smoothed components of the State Space model.

• **diagnostics**: A list containing items useful for diagnostical tests.
• `optim` (if `fit = TRUE`): A list containing the variables that are returned by the `optim` or `optimr` function.

• `loglik_fun`: Function that returns the loglikelihood of the specified State Space model, as a function of its parameters.

• `standard_errors` (if `standard_errors = TRUE`): A list containing the standard errors of the parameters of the State Space model.

For extensive details about the object returned, see `vignette("dictionary","statespacer")`.

**Author(s)**

Dylan Beijers, <dylanbeijers@gmail.com>

**References**


**Examples**

```r
# Fits a local level model for the Nile data
library(datasets)
library(statespacer)

y <- matrix(Nile)
fit <- statespacer(initial = 10, y = y, local_level_ind = TRUE)

# Plots the filtered estimates
plot(1871:1970, fit$function_call$y, type = "p", ylim = c(500, 1400), xlab = NA, ylab = NA)
lines(1871:1970, fit$filtered$level, type = "l")
lines(1871:1970, fit$filtered$level + 1.644854 * sqrt(fit$filtered$P[1, 1, , ]), type = "l", col = "gray")
lines(1871:1970, fit$filtered$level - 1.644854 * sqrt(fit$filtered$P[1, 1, , ]), type = "l", col = "gray")

# Plots the smoothed estimates
plot(1871:1970, fit$function_call$y, type = "p", ylim = c(500, 1400), xlab = NA, ylab = NA)
```

lines(1871:1970, fit$smoothed$level, type = "l")
lines(
  1871:1970, fit$smoothed$level +
  1.644854 * sqrt(fit$smoothed$V[1, 1, ]),
  type = "l", col = "gray"
)
lines(
  1871:1970, fit$smoothed$level -
  1.644854 * sqrt(fit$smoothed$V[1, 1, ]),
  type = "l", col = "gray"
)
Index

as.matrix, 2
BlockMatrix, 2
Cholesky, 3
CoeffARMA, 4
optim, 10, 12
optimr, 10, 12
predict.statespace, 5
SimSmoother, 6
statespace, 5, 6, 7