Package ‘APRScenario’

July 24, 2025

Title Structural Scenario Analysis for Bayesian Structural Vector Autoregression Models

Version 0.0.3.0

Author Giovanni Lombardo [aut, cre]

Maintainer Giovanni Lombardo <giannilmbd@gmail.com>

Description Implements the scenario analysis proposed by Antolin-Diaz, Petrella and Rubio-Ramirez (2021) ``Structural scenario analysis with SVARs" <doi:10.1016/j.jmoneco.2020.06.001>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, lubridate, parallel, MASS, tidyr, abind, psych, Rcpp (>= 1.0.12), RcppProgress

Suggests bsvars, bsvarSIGNs, devtools, ggfortify, gridExtra, zoo, knitr, rmarkdown, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo, RcppProgress

SystemRequirements LAPACK

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-07-24 13:00:07 UTC

Contents

`big_b_and_M` *big_b_and_M This function returns the extended b and M matrices as in APR*

Description

big_b_and_M This function returns the extended b and M matrices as in APR

Usage

```
big_b_and_M(h, n_draws, n_var, n_p, data_ = NULL, matrices = NULL)
```

Arguments

`h`	forecast horison
`n_draws`	Number of draws
`n_var`	Number of variables
`n_p`	Number of lags
`data_`	(matrix optional) The data, stacking Y over X (data and laggs) – columns are observations (default taken from matrices$Z) NB: this is not necessarily the same as the data used to estimate the model If run counterfactuals in previoius historical period (ie not forecast) must pass the data up to previous period relative to counterfactual
`matrices`	Optional matrices object from gen_mats() (default taken from calling environment)

Value

the big_b and big_M matrices of mean and IRF

Examples

```
## Not run:
# Example usage for creating extended matrices
result <- big_b_and_M(h = 4, n_draws = 1000, n_var = 3, n_p = 2,
                      matrices = matrices)
big_b <- result[[1]]
big_M <- result[[2]]

## End(Not run)
```

`forc_h` *forc_h function*

Description

forc_h function

Usage

```
forc_h(h = 1, n_sim = 200, data_ = NULL, posterior = NULL, matrices = NULL)
```

Arguments

`h`	forecast horizon
`n_sim`	length of shock simulation
`data_`	Optional matrix of data n_var*h+1 x T. If NULL, defaults to matrices$Z
`posterior`	Optional posterior object (default taken from calling environment)
`matrices`	Optional matrices object from gen_mats() (default taken from calling environment)

Value

a matrix of unconditional forecasts

Examples

```
## Not run:
# Example usage for unconditional forecasting
forecast <- forc_h(h = 4, n_sim = 1000,
                   posterior = posterior, matrices = matrices)

## End(Not run)
```

`full_scenarios_core` *Exported version of full_scenarios_core*

Description

This function wraps the Rcpp-exported version of `full_scenarios_core` and allows external users to call it with correct argument checks.

Usage

```
full_scenarios_core(
  big_b,
  big_M,
  obs,
  path,
  shocks,
  h,
  n_var,
  g_ = NULL,
  Sigma_g_ = NULL
)
```

Arguments

`big_b`	Cube of B matrices
`big_M`	Cube of M matrices
`obs`	Indices of constrained observables
`path`	Flattened path for observables
`shocks`	Indices of shocks to be recovered
`h`	Forecast horizon
`n_var`	Number of variables
`g_`	Optional vector of non-driving shocks
`Sigma_g_`	Optional covariance matrix of non-driving shocks

Value

A list with elements depending on the input configuration. Typically includes:

mu_eps Matrix of mean structural shocks

Sigma_eps Covariance matrix of structural shocks

mu_y Matrix of conditional means of observables

Sigma_y Covariance matrix of observables

big_b Slice of B matrices used

big_M Slice of M matrices used

draws_used Indices of posterior draws used in the simulation

Examples

```
## Not run:
# This function is typically called internally by scenarios()
# Example usage with simulated data:
big_b <- array(rnorm(9*4*10), dim = c(9, 4, 10))
big_M <- array(rnorm(9*9*10), dim = c(9, 9, 10))
result <- full_scenarios_core(big_b, big_M, obs = 1:2,
                              path = c(1.0, 1.1), shocks = NA,
                              h = 2, n_var = 3)

## End(Not run)
```

`gen_mats` *gen_mats function*

Description

this function returns the matrices necessary for forecasts

Usage

```
gen_mats(posterior = NULL, specification = NULL)
```

Arguments

`posterior`	Posterior estimation results (eg from BsvarSIGNs)
`specification`	Optional specification object (default taken from calling environment)

Value

Returns all objects necessary for scenario analysis (e.g., IRF matrix), including: `M`, `M_inv`, `M_list`, `B`, `B_list`, `n_p`, `n_var`, `Y`, `X`, and `Z`.

Examples

```
library(APRScenario)
data(NKdata)

# Minimal example with a toy specification
spec <- bsvarSIGNs::specify_bsvarSIGN$new(as.matrix(NKdata[,2:4]), p = 1)
est <- bsvars::estimate(spec, S = 10)  # Use small S for fast test
gen_mats(posterior = est, specification = spec)
```

KL *KL function APR suggest this measure to assess the "plausibility" of the conditional forecast. It is based on the Kullback-Leibler measure of distance between the unconditional forecast and the conditional/scenario forecast.*

Description

KL function APR suggest this measure to assess the "plausibility" of the conditional forecast. It is based on the Kullback-Leibler measure of distance between the unconditional forecast and the conditional/scenario forecast.

Usage

```
KL(Sigma_eps, mu_eps, h, plot_ = FALSE, max_cores = NULL)
```

Arguments

Sigma_eps	variance of innovation
mu_eps	mean of innovation
h	forecast horizon
plot_	logical; if TRUE then a histogram of the KL measure is returned
max_cores	maximum number of cores to use for parallel processing (default: NULL, uses CRAN-compliant detection with Windows=1)

Value

Returns the APR ’q’: ie distance from a fair binomial distribution

Examples

```
# Example with simulated innovation data
# Set dimensions
n_var <- 3
h <- 4
n_draws <- 10
n_innovations <- n_var * h

# Create simulated innovation means and covariances
set.seed(123)
mu_eps <- array(rnorm(n_innovations * 1 * n_draws, mean = 0, sd = 0.1),
                dim = c(n_innovations, 1, n_draws))

Sigma_eps <- array(0, dim = c(n_innovations, n_innovations, n_draws))
for (d in 1:n_draws) {
  temp_cov <- matrix(rnorm(n_innovations^2), n_innovations, n_innovations)
  Sigma_eps[,,d] <- temp_cov %*% t(temp_cov) + diag(n_innovations) * 0.5
}

# Calculate KL measure
kl_result <- KL(Sigma_eps, mu_eps, h, plot_ = FALSE)
print(head(kl_result[[1]]))  # Print first few q values
```

`mat_forc` *mat_forc function ## NB: HERE WE USE Antolin-Diaz et al notation # B is reduced form; # A is structural; # d is intercepts # M is reduced so that E(u*u')=Sigma=(A_0*A_0')^(-1) and M_0=A_0^(-1)*Q # Note that the code returns conflicting notation: # B=>A_0^(-1)*Q and # A=>B # ##*

Description

mat_forc function ## NB: HERE WE USE Antolin-Diaz et al notation # B is reduced form; # A is structural; # d is intercepts # M is reduced so that E(u*u')=Sigma=(A_0*A_0')^(-1) and M_0=A_0^(-1)*Q # Note that the code returns conflicting notation: # B=>A_0^(-1)*Q and # A=>B # ##

Usage

```
mat_forc(h = 1, n_draws, n_var, n_p, data_ = NULL, matrices = NULL)
```

Arguments

`h` (integer) forecast horison

`n_draws` (integer) Number of draws

`n_var` (integer) Number of variables

`n_p` (integer) Number of lags

`data_` (matrix optional) The data, stacking Y over X (data and laggs) – columns are observations (default taken from matrices$Z) NB: this is not necessarily the same as the data used to estimate the model If run counterfactuals in previoius historical period (ie not forecast) must pass the data up to previous period relative to counterfactual

`matrices` Optional matrices object from gen_mats() (default taken from calling environment)

Value

the big_b and big_M matrices of mean and IRF

Examples

```
library(APRScenario)
data(NKdata)

# Minimal example with a toy specification
spec <- bsvarSIGNs::specify_bsvarSIGN$new(as.matrix(NKdata[,2:4]), p = 1)
est <- bsvars::estimate(spec, S = 10)  # Use small S for fast test
matrices<-gen_mats(posterior = est, specification = spec)

# Example usage for matrix forecasting
result <- mat_forc(h = 4, n_draws = 10, n_var = 3, n_p = 1,
                   matrices = matrices)
```

NKdata *Example Dataset NKdata*

Description

A dataset used in the APRScenario package.

Usage

```
data(NKdata)
```

Format

A data frame with 244 rows and 4 variables:

GDP.E GDP

pi.PCE.core inflation

i interest rate

year year

plot_bvars *plot_bvars: This function plots the IRFs generated with the BVAR*

Description

plot_bvars: This function plots the IRFs generated with the BVAR

Usage

```
plot_bvars(
  M,
  significance_level = 0.05,
  central_tendency = "mean",
  variable_names = NULL,
  shock_names = NULL
)
```

Arguments

`M`	IRFs produced by eg bvarSIGNs
`significance_level`	(eg 0.05)
`central_tendency`	eg 'mean' or 'median'
`variable_names`	vector of names of variables (strings)
`shock_names`	vector of names of variables (strings)

Value

a list of ggplot objects (plots)

Examples

```
# Example with simulated IRF data
# Create simulated IRF array (n_vars, n_shocks, n_periods, n_draws)
set.seed(123)
n_vars <- 3
n_shocks <- 3
n_periods <- 10
n_draws <- 50

# Generate IRF responses that decay over time
M <- array(0, dim = c(n_vars, n_shocks, n_periods, n_draws))
for (i in 1:n_vars) {
  for (j in 1:n_shocks) {
    for (t in 1:n_periods) {
      # Create decaying responses with some randomness
      base_response <- ifelse(i == j, 1, 0.3) * exp(-0.1 * (t-1))
      M[i, j, t, ] <- rnorm(n_draws, mean = base_response, sd = 0.1)
    }
  }
}

# Create plots
var_names <- c("GDP", "CPI", "FFR")
shock_names <- c("Supply", "Demand", "Monetary")

plots <- plot_bvars(M,
                    variable_names = var_names,
                    shock_names = shock_names,
                    significance_level = 0.1)

# plots is a list of ggplot objects
print(length(plots))
```

---

plot_cond_forc *plot_cond_forc function; Data should conatain the variable "variable", the "hor" horizon and a "history"*

Description

plot_cond_forc function; Data should conatain the variable "variable", the "hor" horizon and a "history"

Usage

```
plot_cond_forc(
  varbl2plot = NULL,
  y_h_cond = NULL,
  center = 0.5,
  lower = 0.16,
  upper = 0.84,
  T.start = NULL,
  T.end = NULL,
  before = 8,
  freq = "quarter",
  y_data = NULL
)
```

Arguments

`varbl2plot`	name of variable to be plotted (string)
`y_h_cond`	conditional forecast data frame (eg from SimScen) with names c("hor","variable","lower","center","upper hor is a Date object
`center`	(optional, default 0.5) quantile of the mid value
`lower`	(optional, default 0.16) quantile of lower range
`upper`	(optional, default 0.84) quantile of upper range
`T.start`	start date of the forecast
`T.end`	end of the forecast
`before`	(integer: optional) periods of data in the plot: default 8 periods
`freq`	(optional, default 'quarter') frequency of the data (eg 'quarter' or 'month')
`y_data`	Data used in the estimation eg t(specification$get_data_matrices()$Y) %>% as.data.frame(); true_data$hor=dates

Value

list of plot and data used

Examples

```
## Not run:
# Example usage with conditional forecast data
plot_result <- plot_cond_forc(varbl2plot = "GDP",
                              y_h_cond = forecast_data,
                              T.start = as.Date("2023-01-01"),
                              T.end = as.Date("2024-01-01"),
                              y_data = historical_data)

## End(Not run)
```

plot_cond_histo *plot_cond_histo function*

Description

This function uses the conditional probability calculations (eg scenarios) and plots the histogram of the selected variable

Usage

```
plot_cond_histo(
  variable = NULL,
  horizon = 1,
  threshold = NULL,
  data = NULL,
  above = TRUE,
  size = 5
)
```

Arguments

`variable`	(character) Name of variable to be plotted
`horizon`	(numeric) At which horizon (horizon<=h)
`threshold`	(numeric,optional) If present compute P(x>threshold)
`data`	data of conditional forecasts
`above`	(logical,optional): if TRUE then compute probability above threshold
`size`	(optional) size of annotation text in the plot

Value

ggplot object (plot)

Examples

```
# Example with simulated conditional forecast data
# Create sample forecast data matrix
set.seed(123)
n_sims <- 500
horizons <- 3
variables <- c("GDP", "CPI", "FFR")

# Create column names in the expected format (variable.horizon)
col_names <- outer(variables, 1:horizons, paste, sep = ".")

# Generate random forecast data
forecast_data <- matrix(rnorm(n_sims * length(col_names)),
                       nrow = n_sims, ncol = length(col_names))
colnames(forecast_data) <- as.vector(col_names)

# Plot histogram for GDP at horizon 2
p <- plot_cond_histo(data = t(forecast_data),
                     variable = "GDP",
                     horizon = 2,
                     threshold = 0.5,
                     above = TRUE)
```

scenarios   *scenarios function (fully optimized with Rcpp) This function computes the mean and covariances to draw from the conditional forecast The actual draw is done in the simscen function*

Description

scenarios function (fully optimized with Rcpp) This function computes the mean and covariances to draw from the conditional forecast The actual draw is done in the simscen function

Usage

```
scenarios(
  h = 3,
  path = NULL,
  obs = NULL,
  free_shocks = NULL,
  n_sample = NULL,
  data_ = NULL,
  g = NULL,
  Sigma_g = NULL,
  posterior = NULL,
  matrices = NULL
)
```

Arguments

`h`	forecast horizon
`path`	conditional path of observables
`obs`	position of observable(s)
`free_shocks`	position of non-driving shocks (NA if all driving)
`n_sample`	Number of draws to sample (<= n_draws)
`data_`	Optional matrix of data (default taken from matrices$Z)
`g`	Optional matrix of non-driving shocks
`Sigma_g`	Optional covariance matrix of non-driving shocks
`posterior`	Optional posterior object (default taken from calling environment)
`matrices`	Optional matrices object from gen_mats() (default taken from calling environment)

Value

list of mu_eps, Sigma_eps, mu_y, Sigma_y, big_b, big_M, draws_used

Examples

```
library(APRScenario)
data(NKdata)

# Minimal example with a toy specification
spec <- bsvarSIGNs::specify_bsvarSIGN$new(as.matrix(NKdata[,2:4]), p = 1)
posterior <- bsvars::estimate(spec, S = 10)  # Use small S for fast test
matrices<-gen_mats(posterior = posterior, specification = spec)
# and having posterior object
 scenario_result <- scenarios(h = 2,
                              path = c(1.0, 1.1),
                              obs = 1,
                              free_shocks = NA,
                              posterior = posterior,
                              matrices = matrices)
```

`SimScen` *simscen function This function takes the mean and covariance of the conditional forecast to draw from the conditional forecast distribution The shock uncertainty is included in the simulation by default, but can be turned off.*

Description

simscen function This function takes the mean and covariance of the conditional forecast to draw from the conditional forecast distribution The shock uncertainty is included in the simulation by default, but can be turned off.

Usage

```
SimScen(
  mu_eps,
  Sigma_eps,
  mu_y,
  Sigma_y,
  big_b,
  big_M,
  n_sim,
  h,
  varbls,
  idx_sampled = 1:dim(mu_eps)[3],
  shock_uncertainty = TRUE
)
```

Arguments

`mu_eps`	mean innovation
`Sigma_eps`	variance innovation
`mu_y`	mean forecast
`Sigma_y`	variance forecast
`big_b`	history forecast
`big_M`	IRF (innovation loading)
`n_sim`	number of simulations
`h`	horizon
`varbls`	variable names
`idx_sampled`	index of random sample to use instead of full draws (from scenarios)
`shock_uncertainty`	(logical; optional) whether to include uncertainty in shocks (default is TRUE)

Value

conditional forecast path and distribution

Examples

```
## Not run:
# Example usage after scenarios() function call
# Requires scenario results from scenarios() function
result <- SimScen(mu_eps = scenario_output$mu_eps,
                  Sigma_eps = scenario_output$Sigma_eps,
                  mu_y = scenario_output$mu_y,
                  Sigma_y = scenario_output$Sigma_y,
                  big_b = scenario_output$big_b,
                  big_M = scenario_output$big_M,
                  n_sim = 1000, h = 3, varbls = c("GDP", "CPI", "FFR"))
```

```
## End(Not run)
```

---

simulate_conditional_forecasts

Simulate paths from conditional forecast distributions

Description

Simulate paths from conditional forecast distributions

Usage

```
simulate_conditional_forecasts(mu_y, Sigma_y, varnames, n_sim = 1000)
```

Arguments

`mu_y`	Array (n_state × 1 × n_draws): conditional forecast mean
`Sigma_y`	Array (n_state × n_state × n_draws): conditional forecast variance
`varnames`	Character vector of variable names (length = number of variables)
`n_sim`	Number of simulations per draw

Value

Array of dimensions (n_state x n_sim x n_draws) of simulated draws with named rows

Note

Users should set their own seed before calling this function if reproducible results are desired.

Examples

```
# Example with simulated data
# Create example data dimensions
n_var <- 3
h <- 2
n_draws <- 5
n_state <- n_var * h

# Simulate conditional forecast means and covariances
set.seed(123)
mu_y <- array(rnorm(n_state * 1 * n_draws), dim = c(n_state, 1, n_draws))
Sigma_y <- array(0, dim = c(n_state, n_state, n_draws))
for (d in 1:n_draws) {
  temp_cov <- matrix(rnorm(n_state^2), n_state, n_state)
  Sigma_y[,,d] <- temp_cov %*% t(temp_cov) + diag(n_state) * 0.1
}
```

```
# Variable names
varnames <- c("GDP", "CPI", "FFR")

# Simulate conditional forecasts
sims <- simulate_conditional_forecasts(mu_y, Sigma_y, varnames, n_sim = 50)
print(dim(sims))
print(rownames(sims)[1:6])
```

Index