Package: sovereign (via r-universe)

August 28, 2024

Title State-Dependent Empirical Analysis

Version 1.2.1

Description A set of tools for state-dependent empirical analysis through both VAR- and local projection-based state-dependent forecasts, impulse response functions, historical decompositions, and forecast error variance decompositions.

License GPL-3

URL https://github.com/tylerJPike/sovereign, https://tylerjpike.github.io/sovereign/

BugReports https://github.com/tylerJPike/sovereign/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Imports broom, dplyr, future, furrr, ggplot2, gridExtra, lmtest, lubridate, magrittr, mclust, purrr, randomForest, sandwich, stats, stringr, strucchange, tidyr, xts, zoo

Suggests testthat, knitr, rmarkdown, quantmod, covr

VignetteBuilder knitr

Repository https://tylerjpike.r-universe.dev

RemoteUrl https://github.com/tylerjpike/sovereign

RemoteRef HEAD

RemoteSha d1257212721bb7e1321777b579c4ec5614d76607

Contents

`covid_volatility_correction`

Lenza-Primiceri Covid Shock Correction

Description

Implement the deterministic volatility correction method of Lenza, Michele and Giorgio Primiceri "How to Estimate a VAR after March 2020" (2020) [NBER Working Paper]. Correction factors are estimated via maximum likelihood.

Usage

```
covid_volatility_correction(var, theta_initial = c(5, 2, 1.5, 0.8))
```

Arguments

`var`	VAR object
`theta_initial`	double: four element vector with scaling parameters, theta in Lenza and Primiceri (2020)

Value

var object

See Also

VAR()

var_irf()

var_fevd()

var_hd()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2018-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)

# estimate VAR
var =
  sovereign::VAR(
    data = Data,
    horizon = 10,
    freq = 'month',
    lag.ic = 'BIC',
    lag.max = 4)

# correct VAR for COVID shock
var = sovereign::covid_volatility_correction(var)

# impulse response functions
var.irf = sovereign::var_irf(var)

# forecast error variance decomposition
var.fevd = sovereign::var_fevd(var)

# historical shock decomposition
var.hd = sovereign::var_hd(var)
```

FEVD *Estimate forecast error variance decomposition*

Description

Estimate the forecast error variance decomposition for VARs with either short or 'IV-short' structural errors. See VAR and RVAR documentation for details regarding structural errors.

Usage

```
FEVD(model, horizon = 10, scale = TRUE)
```

Arguments

model	VAR or RVAR class object
horizon	int: number of periods
scale	boolean: scale variable contribution as percent of total error

Value

long-form data.frame

See Also

VAR()

var_fevd()

RVAR()

rvar_fevd()

Examples

```
 # simple time series
 AA = c(1:100) + rnorm(100)
 BB = c(1:100) + rnorm(100)
 CC = AA + BB + rnorm(100)
 date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
 Data = data.frame(date = date, AA, BB, CC)

 # estimate VAR
  var =
    sovereign::VAR(
      data = Data,
      horizon = 10,
      freq = 'month',
      lag.ic = 'BIC',
      lag.max = 4)

# impulse response functions
var.irf = sovereign::IRF(var)

# forecast error variance decomposition
var.fevd = sovereign::FEVD(var)

# historical shock decomposition
var.hd = sovereign::HD(var)
```

HD *Estimate historical decomposition*

Description

Estimate the historical decomposition for VARs with either 'short' or 'IV-short' structural errors. See VAR and RVAR documentation for details regarding structural errors.

Usage

```
HD(model)
```

Arguments

model	VAR or RVAR class object

Value

long-from data.frame

See Also

VAR()

var_hd()

RVAR()

rvar_hd()

Examples

```
 # simple time series
 AA = c(1:100) + rnorm(100)
 BB = c(1:100) + rnorm(100)
 CC = AA + BB + rnorm(100)
 date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
 Data = data.frame(date = date, AA, BB, CC)

 # estimate VAR
  var =
    sovereign::VAR(
      data = Data,
      horizon = 10,
      freq = 'month',
      lag.ic = 'BIC',
      lag.max = 4)

# impulse response functions
var.irf = sovereign::IRF(var)
```

```
# forecast error variance decomposition
var.fevd = sovereign::FEVD(var)

# historical shock decomposition
var.hd = sovereign::HD(var)
```

IRF *Estimate impulse response functions*

Description

See VAR, RVAR, LP, and RLP documentation for details regarding models and structural errors.

Usage

```
IRF(
  model,
  horizon = 10,
  CI = c(0.1, 0.9),
  bootstrap.type = "auto",
  bootstrap.num = 100,
  bootstrap.parallel = FALSE,
  bootstrap.cores = -1
)
```

Arguments

`model`	VAR, RVAR, LP, or RLP class object
`horizon`	int: number of periods
`CI`	numeric vector: c(lower ci bound, upper ci bound)
`bootstrap.type`	string: bootstrapping technique to use ('auto', 'standard', or 'wild'); if auto then wild is used for IV or IV-short, else standard is used
`bootstrap.num`	int: number of bootstraps
`bootstrap.parallel`	boolean: create IRF draws in parallel
`bootstrap.cores`	int: number of cores to use in parallel processing; -1 detects and uses half the available cores

Value

data frame with columns `target`, `shock`, `horizon`, `response.lower`, `response`, `response.upper`; regime-based models return a list with a data frame per regime.

See Also

var_irf()

rvar_irf()

lp_irf()

rlp_irf()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)

# estimate VAR
 var =
   sovereign::VAR(
     data = Data,
     horizon = 10,
     freq = 'month',
     lag.ic = 'BIC',
     lag.max = 4
   )

# impulse response function
var.irf = sovereign::IRF(var)

 # local projection forecasts
 lp =
   sovereign::LP(
     data = Data,
     horizon = c(1:10),
     lag.ic = 'AIC',
     lag.max = 4,
     type =  'both',
     freq = 'month')

 # LP impulse response function
 lp.irf = sovereign::IRF(lp)
```

Description

Estimate local projections

Usage

```
LP(
  data,
  horizons = 1,
  freq = "month",
  type = "const",
  p = 1,
  lag.ic = NULL,
  lag.max = NULL,
  NW = FALSE,
  NW_lags = NULL,
  NW_prewhite = NULL
)
```

Arguments

`data`	data.frame, matrix, ts, xts, zoo: Endogenous regressors
`horizons`	int: forecast horizons
`freq`	string: frequency of data ('day', 'week', 'month', 'quarter', or 'year')
`type`	string: type of deterministic terms to add ('none', 'const', 'trend', or 'both')
`p`	int: lags
`lag.ic`	string: information criterion to choose the optimal number of lags ('AIC' or 'BIC')
`lag.max`	int: maximum number of lags to test in lag selection
`NW`	boolean: Newey-West correction on variance-covariance matrix
`NW_lags`	int: number of lags to use in Newey-West correction
`NW_prewhite`	boolean: TRUE prewhite option for Newey-West correction (see sandwich::NeweyWest)

Value

list object with elements `data`, `model`, `forecasts`, `residuals`; if there is more than one forecast horizon estimated, then `model`, `forecasts`, `residuals` will each be a list where each element corresponds to a single horizon

References

1. Jorda, Oscar "Estimation and Inference of Impulse Responses by Local Projections" 2005.

See Also

LP()

lp_irf()

RLP()

rlp_irf()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)

# local projection forecasts
lp =
  sovereign::LP(
    data = Data,
    horizon = c(1:10),
    lag.ic = 'AIC',
    lag.max = 4,
    type =  'both',
    freq = 'month')

# impulse response function
irf = sovereign::lp_irf(lp)
```

---

lp_irf *Estimate impulse response functions*

Description

Estimate impulse response functions

Usage

```
lp_irf(lp, CI = c(0.1, 0.9), regime = NULL)
```

Arguments

lp	LP output
CI	numeric vector: c(lower ci bound, upper ci bound)
regime	string: indicates regime index column of data

Value

long-form data.frame with one row per target-shock-horizon identifier

See Also

LP()

lp_irf()

RLP()

rlp_irf()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)

# local projection forecasts
lp =
  sovereign::LP(
    data = Data,
    horizon = c(1:10),
    lag.ic = 'AIC',
    lag.max = 4,
    type =  'both',
    freq = 'month')

# impulse response function
irf = sovereign::lp_irf(lp)
```

---

plot_error *Chart residuals*

Description

Chart residuals

Usage

```
plot_error(residuals, series = NULL, verticle = FALSE)
```

Arguments

residuals	data.frame: sovereign residuals object
series	string: series to plot (default to all series)
verticle	boolean: If true then stack all plots into one column

Value

grid of ggplot2 graphs

---

plot_fevd *Chart FEVDs*

Description

Chart FEVDs

Usage

```
plot_fevd(fevd, responses = NULL, verticle = FALSE)
```

Arguments

fevd	fevd object
responses	string vector: responses to plot
verticle	boolean: If true then stack all plots into one column

Value

grid of ggplot2 graphs

---

plot_forecast *Chart forecasts*

Description

Chart forecasts

Usage

```
plot_forecast(forecasts, series = NULL, verticle = FALSE)
```

Arguments

forecasts	data.frame: sovereign forecast object
series	string: series to plot (default to all series)
verticle	boolean: If true then stack all plots into one column

Value

grid of ggplot2 graphs

---

plot_hd *Chart HDs*

Description

Chart HDs

Usage

```
plot_hd(hd, verticle = FALSE)
```

Arguments

hd	hd object
verticle	boolean: If true then stack all plots into one column

Value

grid of ggplot2 graphs

---

plot_individual_error *Chart individual residuals*

Description

Chart individual residuals

Usage

```
plot_individual_error(
  data,
  target,
  title = NULL,
  ylab = NULL,
  freq = NULL,
  zeroline = FALSE
)
```

Arguments

`data`	data.frame: sovereign residuals object
`target`	string: series to plot
`title`	string: chart title
`ylab`	string: y-axis label
`freq`	string: frequency (acts as sub-title)
`zeroline`	boolean: if TRUE then add a horizontal line at zero

Value

ggplot2 chart

`plot_individual_fevd` *Plot an individual FEVD*

Description

Plot an individual FEVD

Usage

```
plot_individual_fevd(fevd, response.var, title, ylab)
```

Arguments

`fevd`	fevd object
`response.var`	string: name of variable to treat as the response
`title`	string: title of the chart
`ylab`	string: y-axis label

Value

ggplot2 graph

plot_individual_forecast *Chart individual forecast*

Description

Chart individual forecast

Usage

```
plot_individual_forecast(
  data,
  target,
  title = NULL,
  ylab = NULL,
  freq = NULL,
  zeroline = FALSE
)
```

Arguments

data	data.frame: sovereign model forecast
target	string: series to plot
title	string: chart title
ylab	string: y-axis label
freq	string: frequency (acts as sub-title)
zeroline	boolean: if TRUE then add a horizontal line at zero

Value

ggplot2 chart

plot_individual_hd *Plot an individual HD*

Description

Plot an individual HD

Usage

```
plot_individual_hd(hd, target.var, title)
```

Arguments

hd	hd object
target.var	string: name of variable to decompose into shocks
title	string: title of the chart

Value

ggplot2 graph

---

plot_individual_irf *Plot an individual IRF*

Description

Plot an individual IRF

Usage

```
plot_individual_irf(irf, shock.var, response.var, title, ylab)
```

Arguments

irf	irf object
shock.var	string: name of variable to treat as the shock
response.var	string: name of variable to treat as the response
title	string: title of the chart
ylab	string: y-axis label

Value

ggplot2 graph

plot_irf *Chart IRFs*

Description

Chart IRFs

Usage

```
plot_irf(irf, shocks = NULL, responses = NULL, verticle = FALSE)
```

Arguments

irf	irf object
shocks	string vector: shocks to plot
responses	string vector: responses to plot
verticle	boolean: If true then stack all plots into one column

Value

grid of ggplot2 graphs

regimes *Identify regimes via unsupervised ML algorithms*

Description

Regime assignment (clustering) methods available include the unsupervised random forest, k-mean clustering, Fraley and Raftery Model-based clustering EM algorithm, and the Bai & Perron (2003) method for simultaneous estimation of multiple breakpoints.

Usage

```
regimes(data, method = "rf", regime.n = NULL)
```

Arguments

data	data.frame, matrix, ts, xts, zoo: Endogenous regressors
method	string: regime assignment technique ('rf', 'kmeans', 'EM', or 'BP)
regime.n	int: number of regimes to estimate (applies to kmeans and EM)

Value

data as a data.frame with a regime column assigning rows to mutually exclusive regimes

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)

# estimate reigme
regime =
 sovereign::regimes(
    data = Data,
    method = 'kmeans',
    regime.n = 3)
```

RLP *Estimate regime-dependent local projections*

Description

Estimate a regime-dependent local projection (i.e. a state-dependent LP), with an exogenous state indicator, of the specification:

$$Y_{t+h} = X_t \beta_{s_t} + \epsilon_t$$

where *t* is the time index, and *s* is a mutually exclusive state of the world observed at time *t*. When the regime vector is not supplied by the user, then a two-state regime series is estimated via random forest.

Usage

```
RLP(
  data,
  horizons = 1,
  freq = "month",
  type = "const",
  p = 1,
  lag.ic = NULL,
  lag.max = NULL,
  NW = FALSE,
  NW_lags = NULL,
  NW_prewhite = NULL,
  regime = NULL,
  regime.method = "rf",
  regime.n = 2
)
```

Arguments

data	data.frame, matrix, ts, xts, zoo: Endogenous regressors
horizons	int: forecast horizons
freq	string: frequency of data ('day', 'week', 'month', 'quarter', or 'year')
type	string: type of deterministic terms to add ('none', 'const', 'trend', or 'both')
p	int: lags
lag.ic	string: information criterion to choose the optimal number of lags ('AIC' or 'BIC')
lag.max	int: maximum number of lags to test in lag selection
NW	boolean: Newey-West correction on variance-covariance matrix
NW_lags	int: number of lags to use in Newey-West correction
NW_prewhite	boolean: TRUE prewhite option for Newey-West correction (see sandwich::NeweyWest)
regime	string: name or regime assignment vector in the design matrix (data)
regime.method	string: regime assignment technique ('rf', 'kmeans', 'EM', 'BP')
regime.n	int: number of regimes to estimate (applies to kmeans and EM)

Value

list of lists, one list per regime, each regime with objects with elements `data`, `model`, `forecasts`, `residuals`; if there is more than one forecast horizon estimated, then `model`, `forecasts`, `residuals` will each be a list where each element corresponds to a single horizon

References

1. Jorda, Oscar "Estimation and Inference of Impulse Responses by Local Projections" 2005.

See Also

`LP()`

`lp_irf()`

`RLP()`

`rlp_irf()`

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)
# add regime
Data = dplyr::mutate(Data, reg = dplyr::if_else(AA > median(AA), 1, 0))
```

```
# local projection forecasts
rlp =
  sovereign::RLP(
    data = Data,
    regime = 'reg',
    horizon = c(1:10),
    freq = 'month',
    p = 1,
    type =  'const',
    NW = TRUE,
    NW_lags = 1,
    NW_prewhite = FALSE)

# impulse response function
rirf = sovereign::rlp_irf(rlp)
```

`rlp_irf` *Estimate regime-dependent impulse response functions*

Description

Estimate regime-dependent impulse response functions

Usage

```
rlp_irf(rlp, CI = c(0.1, 0.9))
```

Arguments

`rlp`	RLP output
`CI`	numeric vector: c(lower ci bound, upper ci bound)

Value

list of long-form data.frame with one row per target-shock-horizon identifier

See Also

`LP()`

`lp_irf()`

`RLP()`

`rlp_irf()`

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)
# add regime
Data = dplyr::mutate(Data, reg = dplyr::if_else(AA > median(AA), 1, 0))

# local projection forecasts
rlp =
  sovereign::RLP(
    data = Data,
    regime = 'reg',
    horizon = c(1:10),
    freq = 'month',
    p = 1,,
    type =  'const',
    NW = TRUE,
    NW_lags = 1,
    NW_prewhite = FALSE)

# impulse response function
rirf = sovereign::rlp_irf(rlp)
```

RVAR *Estimate regime-dependent VAR, SVAR, or Proxy-SVAR*

Description

Estimate a regime-dependent VAR (i.e. a state-dependent VAR), with an exogenous state indicator, of the specification:

$$Y_{t+1} = X_t \beta_{s_t} + \epsilon_t$$

where *t* is the time index, *Y* is the set of outcome vectors, *X* the design matrix (of *p* lagged values of Y), and *s* is a mutually exclusive state of the world observed at time *t*. When the regime vector is not supplied by the user, then a two-state regime series is estimated via random forest.

Usage

```
RVAR(
  data,
  horizon = 10,
  freq = "month",
  type = "const",
```

```
  p = 1,
  lag.ic = NULL,
  lag.max = NULL,
  regime = NULL,
  regime.method = "rf",
  regime.n = 2,
  structure = "short",
  instrument = NULL,
  instrumented = NULL
)
```

Arguments

data	data.frame, matrix, ts, xts, zoo: Endogenous regressors
horizon	int: forecast horizons
freq	string: frequency of data ('day', 'week', 'month', 'quarter', or 'year')
type	string: type of deterministic terms to add ('none', 'const', 'trend', or 'both')
p	int: lags
lag.ic	string: information criterion to choose the optimal number of lags ('AIC' or 'BIC')
lag.max	int: maximum number of lags to test in lag selection
regime	string: name or regime assignment vector in the design matrix (data)
regime.method	string: regime assignment technique ('rf', 'kmeans', 'EM', or 'BP')
regime.n	int: number of regimes to estimate (applies to kmeans and EM)
structure	string: type of structural identification strategy to use in model analysis (NA, 'short', 'IV', or 'IV-short')
instrument	string: name of instrumental variable contained in the data matrix
instrumented	string: name of variable to be instrumented in IV and IV-short procedure; default is the first non-date variable in data

Details

The regime-dependent VAR is a generalization of the popular threshold VAR - which trades off estimating a threshold value for an endogenous variable for accepting an exogenous regime that can be based on information from inside or outside of the system, with or without parametric assumptions, and with or without timing restrictions. Moreover, the RVAR may be extended to include structural shocks, including the use of instrumental variables.

State dependence. The RVAR augments the traditional VAR by allowing state-dependence in the coefficient matrix. The RVAR differs from the more common threshold VAR (TVAR), due to the fact that states are exegonesouly determined. As a result, the states (i.e. regimes) do not need to be based on information inside the model, moreover, regimes can be determined by any process the user determines best fits their needs. For example, regimes based on NBER dated recessions and expansions are based on a judgmental process considering hundreds of series, potentially none of which are in the VAR being modeled. Alternatively, a user may use unsupervised machine learning to assign regimes - this is the process the `sovereign::regimes` function facilitates.

Structural shocks. See Sims (1980) for details regarding the baseline vector-autoregression (VAR) model. The VAR may be augmented to become a structural VAR (SVAR) with one of three different structural identification strategies:

1. short-term impact restrictions via Cholesky decomposition, see Christiano et al (1999) for details **(structure = 'short')**
2. external instrument identification, i.e. a Proxy-SVAR strategy, see Mertens and Ravn (2013) for details **(structure = 'IV')**
3. or a combination of short-term and IV identification via Lunsford (2015) **(structure = 'IV-short')**

Note that including structure does not change the estimation of model coefficients or forecasts, but does change impulse response functions, forecast error variance decomposition, and historical decompositions. Historical decompositions will not be available for models using the 'IV' structure. Additionally note that only one instrument may be used in this estimation routine.

Value

List of lists, where each regime is a list with items:

1. data: data.frame with endogenous variables and 'date' column.
2. model: list with data.frame of model coefficients (in psuedo-companion form), data.frame of coefficient standard errors, integer of lags p, integer of horizons, string of frequency, string of deterministic term type, numeric of log-likelihood, regime indicator
3. forecasts: list of data.frames per horizon; data.frame with column for date (day the forecast was made), forecast.date (the date being forecasted), target (variable forecasted), and forecast
4. residuals: list of data.frames per horizon; data.frame of residuals
5. structure: string denoting which structural identification strategy will be used in analysis (or NA)
6. instrument: data.frame with 'date' column and 'instrument' column (or NULL)
7. instrumented: string denoting which column will be instrumted in 'IV' and 'IV-short' strategies (or NULL)

References

1. Christiano, Lawrence, Martin Eichenbaum, and Charles Evans "Monetary policy shocks: What have we learned and to what end?" Handbook of Macroeconomics, Vol 1, Part A, 1999.
2. Lunsford, Kurt "Identifying Structural VARs with a Proxy Variable and a Test for a Weak Proxy" 2015.
3. Mertens, Karel and Morten Ravn "The Dynamic Effects of Personal and Corporate Income Tax Changes in the United States" 2013.
4. Sims, Christopher "Macroeconomics and Reality" 1980.

See Also

VAR()

RVAR()

IRF()

FEVD()

HD()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)
Data = dplyr::mutate(Data, reg = dplyr::if_else(AA > median(AA), 1, 0))

# estimate regime-dependent VAR
 rvar =
   sovereign::RVAR(
     data = Data,
     horizon = 10,
     freq = 'month',
     regime.method = 'rf',
     regime.n = 2,
     lag.ic = 'BIC',
     lag.max = 4)

# impulse response functions
rvar.irf = sovereign::rvar_irf(rvar)

# forecast error variance decomposition
rvar.fevd = sovereign::rvar_fevd(rvar)

# historical shock decomposition
rvar.hd = sovereign::rvar_hd(rvar)
```

`rvar_fevd` *Estimate regime-dependent forecast error variance decomposition*

Description

Estimate forecast error variance decomposition for RVARs with either short or 'IV-short' structural errors.

Usage

```
rvar_fevd(rvar, horizon = 10, scale = TRUE)
```

Arguments

rvar	RVAR output
horizon	int: number of periods
scale	boolean: scale variable contribution as percent of total error

Value

list, each regime returns its own long-form data.frame

See Also

VAR()

var_irf()

var_fevd()

var_hd()

RVAR()

rvar_irf()

rvar_fevd()

rvar_hd()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)
Data = dplyr::mutate(Data, reg = dplyr::if_else(AA > median(AA), 1, 0))

# estimate VAR
 rvar =
   sovereign::RVAR(
     data = Data,
     horizon = 10,
     freq = 'month',
     regime.method = 'rf',
     regime.n = 2,
     lag.ic = 'BIC',
     lag.max = 4)

# impulse response functions
rvar.irf = sovereign::rvar_irf(rvar)
```

```
# forecast error variance decomposition
rvar.fevd = sovereign::rvar_fevd(rvar)

# historical shock decomposition
rvar.hd = sovereign::rvar_hd(rvar)
```

---

rvar_hd *Estimate regime-dependent historical decomposition*

Description

Estimate historical decomposition for RVARs with either short or 'IV-short' structural errors.

Usage

```
rvar_hd(rvar)
```

Arguments

rvar	RVAR output

Value

long form data.frames

See Also

VAR()

var_irf()

var_fevd()

var_hd()

RVAR()

rvar_irf()

rvar_fevd()

rvar_hd()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)
Data = dplyr::mutate(Data, reg = dplyr::if_else(AA > median(AA), 1, 0))

# estimate VAR
 rvar =
   sovereign::RVAR(
     data = Data,
     horizon = 10,
     freq = 'month',
     regime.method = 'rf',
     regime.n = 2,
     lag.ic = 'BIC',
     lag.max = 4)

# impulse response functions
rvar.irf = sovereign::rvar_irf(rvar)

# forecast error variance decomposition
rvar.fevd = sovereign::rvar_fevd(rvar)

# historical shock decomposition
rvar.hd = sovereign::rvar_hd(rvar)
```

`rvar_irf` *Estimate regime-dependent impulse response functions*

Description

Estimate regime-dependent impulse response functions

Usage

```
rvar_irf(
  rvar,
  horizon = 10,
  CI = c(0.1, 0.9),
  bootstrap.type = "auto",
  bootstrap.num = 100,
  bootstrap.parallel = FALSE,
```

```
  bootstrap.cores = -1
)
```

Arguments

rvar	RVAR output
horizon	int: number of periods
CI	numeric vector: c(lower ci bound, upper ci bound)
bootstrap.type	string: bootstrapping technique to use ('auto', 'standard', or 'wild'); if auto then wild is used for IV or IV-short, else standard is used
bootstrap.num	int: number of bootstraps
bootstrap.parallel	boolean: create IRF draws in parallel
bootstrap.cores	int: number of cores to use in parallel processing; -1 detects and uses half the available cores

Value

list of regimes, each with data.frame of columns target, shock, horizon, response.lower, response, response.upper

See Also

VAR()

var_irf()

var_fevd()

RVAR()

rvar_irf()

rvar_fevd()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)
Data = dplyr::mutate(Data, reg = dplyr::if_else(AA > median(AA), 1, 0))

# estimate VAR
 rvar =
   sovereign::RVAR(
     data = Data,
     horizon = 10,
```

```
    freq = 'month',
    regime.method = 'rf',
    regime.n = 2,
    lag.ic = 'BIC',
    lag.max = 4)

# impulse response functions
rvar.irf = sovereign::rvar_irf(rvar)

# forecast error variance decomposition
rvar.fevd = sovereign::rvar_fevd(rvar)

# historical shock decomposition
rvar.hd = sovereign::rvar_hd(rvar)
```

VAR *Estimate VAR, SVAR, or Proxy-SVAR*

Description

Estimate VAR, SVAR, or Proxy-SVAR

Usage

```
VAR(
  data,
  horizon = 10,
  freq = "month",
  type = "const",
  p = 1,
  lag.ic = NULL,
  lag.max = NULL,
  structure = "short",
  instrument = NULL,
  instrumented = NULL
)
```

Arguments

data	data.frame, matrix, ts, xts, zoo: Endogenous regressors
horizon	int: forecast horizons
freq	string: frequency of data ('day', 'week', 'month', 'quarter', or 'year')
type	string: type of deterministic terms to add ('none', 'const', 'trend', or 'both')
p	int: lags

`lag.ic`	string: information criterion to choose the optimal number of lags ('AIC' or 'BIC')
`lag.max`	int: maximum number of lags to test in lag selection
`structure`	string: type of structural identification strategy to use in model analysis (NA, 'short', 'IV', or 'IV-short')
`instrument`	string: name of instrumental variable contained in the data matrix
`instrumented`	string: name of variable to be instrumented in IV and IV-short procedure; default is the first non-date variable in data

Details

See Sims (1980) for details regarding the baseline vector-autoregression (VAR) model. The VAR may be augmented to become a structural VAR (SVAR) with one of three different structural identification strategies:

1. short-term impact restrictions via Cholesky decomposition, see Christiano et al (1999) for details **(structure = 'short')**
2. external instrument identification, i.e. a Proxy-SVAR strategy, see Mertens and Ravn (2013) for details **(structure = 'IV')**
3. or a combination of short-term and IV identification via Lunsford (2015) **(structure = 'IV-short')**

Note that including structure does not change the estimation of model coefficients or forecasts, but does change impulse response functions, forecast error variance decomposition, and historical decompositions. Historical decompositions will not be available for models using the 'IV' structure. Additionally note that only one instrument may be used in this estimation routine.

Value

1. data: data.frame with endogenous variables and 'date' column.
2. model: list with data.frame of model coefficients (in psuedo-companion form), data.frame of coefficient standard errors, integer of lags p, integer of horizons, string of frequency, string of deterministic term type, numeric of log-likelihood
3. forecasts: list of data.frames per horizon; data.frame with column for date (day the forecast was made), forecast.date (the date being forecasted), target (variable forecasted), and forecast
4. residuals: list of data.frames per horizon; data.frame of residuals
5. structure: string denoting which structural identification strategy will be used in analysis (or NA)
6. instrument: data.frame with 'date' column and 'instrument' column (or NULL)
7. instrumented: string denoting which column will be instrumted in 'IV' and 'IV-short' strategies (or NA)

References

1. Christiano, Lawrence, Martin Eichenbaum, and Charles Evans "Monetary policy shocks: What have we learned and to what end?" Handbook of Macroeconomics, Vol 1, Part A, 1999.
2. Lunsford, Kurt "Identifying Structural VARs with a Proxy Variable and a Test for a Weak Proxy" 2015.
3. Mertens, Karel and Morten Ravn "The Dynamic Effects of Personal and Corporate Income Tax Changes in the United States" 2013.
4. Sims, Christopher "Macroeconomics and Reality" 1980.

See Also

VAR()

var_irf()

var_fevd()

var_hd()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)

# estimate VAR
 var =
   sovereign::VAR(
     data = Data,
     horizon = 10,
     freq = 'month',
     lag.ic = 'BIC',
     lag.max = 4)

 # impulse response functions
 var.irf = sovereign::var_irf(var)

 # forecast error variance decomposition
 var.fevd = sovereign::var_fevd(var)

 # historical shock decomposition
 var.hd = sovereign::var_hd(var)
```

var_fevd *Estimate forecast error variance decomposition*

Description

Estimate forecast error variance decomposition for VARs with either short or 'IV-short' structural errors.

Usage

```
var_fevd(var, horizon = 10, scale = TRUE)
```

Arguments

var	VAR output
horizon	int: number of periods
scale	boolean: scale variable contribution as percent of total error

Value

long-form data.frame

See Also

VAR()

var_irf()

var_fevd()

var_hd()

RVAR()

rvar_irf()

rvar_fevd()

rvar_hd()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)

# estimate VAR
 var =
   sovereign::VAR(
```

```
        data = Data,
        horizon = 10,
        freq = 'month',
        lag.ic = 'BIC',
        lag.max = 4)

# impulse response functions
var.irf = sovereign::var_irf(var)

# forecast error variance decomposition
var.fevd = sovereign::var_fevd(var)

# historical shock decomposition
var.hd = sovereign::var_hd(var)
```

---

var_hd *Estimate historical decomposition*

Description

Estimate historical decomposition for VARs with either short or 'IV-short' structural errors.

Usage

```
var_hd(var)
```

Arguments

var	VAR output

Value

long-from data.frame

See Also

VAR()

var_irf()

var_fevd()

var_hd()

RVAR()

rvar_irf()

rvar_fevd()

rvar_hd()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)

# estimate VAR
 var =
   sovereign::VAR(
     data = Data,
     horizon = 10,
     freq = 'month',
     lag.ic = 'BIC',
     lag.max = 4)

# impulse response functions
var.irf = sovereign::var_irf(var)

# forecast error variance decomposition
var.fevd = sovereign::var_fevd(var)

# historical shock decomposition
var.hd = sovereign::var_hd(var)
```

---

var_irf *Estimate impulse response functions*

Description

Estimate impulse response functions

Usage

```
var_irf(
  var,
  horizon = 10,
  CI = c(0.1, 0.9),
  bootstrap.type = "auto",
  bootstrap.num = 100,
  bootstrap.parallel = FALSE,
  bootstrap.cores = -1
)
```

Arguments

var	VAR output
horizon	int: number of periods
CI	numeric vector: c(lower ci bound, upper ci bound)
bootstrap.type	string: bootstrapping technique to use ('auto', 'standard', or 'wild'); if auto then wild is used for IV or IV-short, else standard is used
bootstrap.num	int: number of bootstraps
bootstrap.parallel	boolean: create IRF draws in parallel
bootstrap.cores	int: number of cores to use in parallel processing; -1 detects and uses half the available cores

Value

data.frame with columns target, shock, horizon, response.lower, response, response.upper

See Also

VAR()

var_irf()

var_fevd()

var_hd()

RVAR()

rvar_irf()

rvar_fevd()

rvar_hd()

Examples

```
# simple time series
AA = c(1:100) + rnorm(100)
BB = c(1:100) + rnorm(100)
CC = AA + BB + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, AA, BB, CC)

# estimate VAR
 var =
   sovereign::VAR(
     data = Data,
     horizon = 10,
     freq = 'month',
     lag.ic = 'BIC',
     lag.max = 4)
```

```
# impulse response functions
var.irf = sovereign::var_irf(var)

# forecast error variance decomposition
var.fevd = sovereign::var_fevd(var)

# historical shock decomposition
var.hd = sovereign::var_hd(var)
```

Index