Package 'factorstochvol'

December 31, 2016

Encoding UTF-8

Type Package

Title Bayesian Estimation of (Sparse) Latent Factor Stochastic Volatility Models

Version 0.8.3

Maintainer Gregor Kastner <gregor.kastner@wu.ac.at>

Description Markov chain Monte Carlo (MCMC) sampler for fully Bayesian estimation of latent factor stochastic volatility models. Sparsity can be achieved through the usage of Normal-Gamma priors on the factor loading matrix.

License GPL (>= 2)

Depends R (>= 3.0.2)

Imports stochvol (>= 1.3.2), GIGrvg (>= 0.4), Rcpp (>= 0.12.7), corrplot, methods, grDevices, graphics, stats, utils

LinkingTo Rcpp, RcppArmadillo (>= 0.7.500.0.0), stochvol

RoxygenNote 5.0.1

NeedsCompilation yes

Author Gregor Kastner [aut, cre]

Repository CRAN

Date/Publication 2016-12-31 17:21:54

R topics documented:

`factorstochvol-package`

Bayesian Estimation of (Sparse) Latent Factor Stochastic Volatility Models through MCMC

Description

This packages provides a Markov chain Monte Carlo (MCMC) sampler for fully Bayesian estimation of latent factor stochastic volatility models. Sparsity can be achieved through the usage of Normal-Gamma priors on the factor loadings matrix.

Details

In recent years, multivariate factor stochastic volatility (SV) models have been increasingly used to analyze financial and economic time series because they can capture joint (co-)volatility dynamics by a small number of latent time-varying factors. The main advantage of such a model is its parsimony, as all variances and covariances of a time series vector are governed by a low-dimensional common factor with the components following independent SV models. For problems of this kind, MCMC is a very efficient estimation method, it is however associated with a considerable computational burden when the number of assets is moderate to large. To overcome this, the latent volatility states are drawn "all without a loop" (AWOL), ancillarity-sufficiency interweaving strategies (ASIS) are applied to sample the univariate components as well as the factor loadings. Thus, this package can be applied directly estimate time-varying covariance and correlation matrices for medium-and high-dimensional time series. To guarantee sparsity, a hierarchical Normal-Gamma prior can be used for the factor loadings matrix which shrinks the unnecessary factor loadings towards zero.

Note

This package is currently in active development; the interface of some of the functions might change. Moreover, even though I tried to carefully check everything, factorstochvol may still contain typos, inconsistencies, or even bugs. Your comments and suggestions are warmly welcome!

Author(s)

Gregor Kastner <gregor.kastner@wu.ac.at>

References

Kastner, G., Frühwirth-Schnatter, S., Lopes, H. F. (2016). Efficient Bayesian inference for multivariate factor stochastic volatility models. *Report 128, Research Report Series of the Institute of Statistics and Mathematics, WU Vienna University of Economics and Business*, http://epub.wu.ac.at/4875/

Kastner, G. (2016). Sparse Bayesian time-varying covariance estimation in many dimensions. *Working Paper*

Kastner, G. and Frühwirth-Schnatter, S. (2014). Ancillarity-sufficiency interweaving strategy (ASIS) for boosting MCMC estimation of stochastic volatility models. *Computational Statistics and Data Analysis*, http://dx.doi.org/10.1016/j.csda.2013.01.002.

See Also

stochvol

Examples

```
## Not run:
set.seed(1)

# simulate data from a (small) factor SV model:
sim <- fsvsim(series = 5, factors = 2)

# estimate the model (CAVEAT: only few draws!)
```

```
res <- fsvsample(sim$y, factors = 2, draws = 2000,
                 burnin = 500, runningstore = 6)

# plot implied volas overtime:
voltimeplot(res)

# plot correlation matrix at some points in time:
par(mfrow = c(2,2))
corimageplot(res, seq(1, nrow(sim$y), length.out = 4),
             fsvsimobj = sim, plotCI = 'circle',
             plotdatedist = -2)


# plot (certain) covariances and correlations over time
par(mfrow = c(2,1))
covtimeplot(res, 1)
cortimeplot(res, 1)

# plot (all) correlations over time
corplot(res, fsvsimobj = sim, these = 1:10)

# plot factor loadings
par(mfrow = c(1,1))
facloadpointplot(res, fsvsimobj = sim)
facloadpairplot(res)
facloadcredplot(res)
facloaddensplot(res, fsvsimobj = sim)

# plot latent log variances
logvartimeplot(res, fsvsimobj = sim, show = "fac")
logvartimeplot(res, fsvsimobj = sim, show = "idi")

# plot communalities over time
comtimeplot(res, fsvsimobj = sim, show = 'joint')
comtimeplot(res, fsvsimobj = sim, show = 'series')

## End(Not run)
```

`comtimeplot` *Plot comunalities over time.*

Description

`comtimeplot` plots the communalities over time, i.e. the series-specific proportion of variance explained through the common factors.

Usage

```
comtimeplot(x, fsvsimobj = NULL, show = "series", maxrows = 5,
  ylim = c(0, 1))
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
fsvsimobj	Object of class 'fsvsim' (or NULL), usually resulting from a call to fsvsim. Defaults to NULL.
show	Indicator whether to show joint ('joint'), series-specific ('series'), or both ('both') communalities.
maxrows	Single positive integer denoting the maximum number of series in each plot. Defaults to 5.
ylim	Vector of length two denoting the range of the horizontal axis. Defaults to 1.

Details

This function displays the joint (average) communalities over time and all series-specific communalities. If communalities haven't been stored during sampling, comtimeplot produces an error.

Value

Returns x invisibly.

See Also

Other plotting: corimageplot, corplot, cortimeplot, facloadcredplot, facloaddensplot, facloadpairplot, facloadpointplot, facloadtraceplot, logvartimeplot, paratraceplot, plot.fsvdraws, voltimeplot

---

corelement *Extract "true" model-implied correlations of two series only*

Description

corelement extracts the model-implied (time-varying) correlations between (exactly) two component series.

Usage

```
corelement(x, i, j, these = seq_len(nrow(x$y)))
```

Arguments

x	Object of class 'fsvsim', usually resulting from a call of the function fsvsim.
i	Index of component series 1.
j	Index of component series 2.
these	Vector indicating which points in time should be extracted.

Value

Vector with the requested correlations.

See Also

Other simulation: covelement, covmat.fsvsim

---

corimageplot *Plot correlation matrices for certain points in time*

Description

corimageplot plots the model-implied correlation matrices for one or several points in time.

Usage

```
corimageplot(x, these = seq_len(nrow(x$y)), order = "original",
  plotdatedist = 0, plotCI = "n", date.cex = 1.5, col = NULL,
  fsvsimobj = NULL, plottype = "corrplot", ...)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
these	Index vector containing the time points to plot. Defaults to seq_len(nrow(x$y)).
order	String, where 'none' and 'original' indicate not to mess with the series ordering. Other keywords (e.g. 'hclust') will be forwarded to corrMatOrder.
plotdatedist	Numerical value indicating where the dates should be plotted.
plotCI	String. If not equal to 'n', posterior credible regions are added (posterior mean +/- 2 posterior sd). Ignored if plottype is "imageplot".
date.cex	Size multiplier for the dates.
col	Color palette or NULL (the default).
fsvsimobj	To indicate data generating values in case of simulated data, pass an object of type fsvsim (usually the result of a call to fsvsim).
plottype	Indicates which type of plot should be drawn. Can be "corrplot" for corrplot (recommended for up to around 20 series), or "imageplot" for a simpler image plot.
...	Additional parameters will be passed on to corrplot. Ignored if plottype is "imageplot".

Value

Returns x invisibly.

Note

If correlations haven't been stored during sampling, `corimageplot` produces an error.

See Also

Other plotting: `comtimeplot`, `corplot`, `cortimeplot`, `facloadcredplot`, `facloaddensplot`, `facloadpairplot`, `facloadpointplot`, `facloadtraceplot`, `logvartimeplot`, `paratraceplot`, `plot.fsvdraws`, `voltimeplot`

`corplot` *Plots pairwise correlations over time*

Description

`corplot` gives an overview of (certain) pairwise correlations. Throws a warning if these haven't been stored during sampling.

Usage

```
corplot(x, fsvsimobj = NULL, these = 1:(ncol(x$y) * (ncol(x$y) - 1)/2),
  start = 1, end = nrow(x$y), maxrows = 10, ...)
```

Arguments

`x`	Object of class `'fsvdraws'`, usually resulting from a call to `fsvsample`.
`fsvsimobj`	To indicate data generating values in case of simulated data, pass an object of type `fsvsim` (usually the result of a call to `fsvsim`).
`these`	Indicator which correlations should be plotted. Default is all.
`start`	First point in time to plot.
`end`	Last point in time to plot.
`maxrows`	The maximum number of rows per page.
`...`	Other arguments will be passed on to `ts.plot`.

Value

Returns `x` invisibly.

See Also

Other plotting: `comtimeplot`, `corimageplot`, `cortimeplot`, `facloadcredplot`, `facloaddensplot`, `facloadpairplot`, `facloadpointplot`, `facloadtraceplot`, `logvartimeplot`, `paratraceplot`, `plot.fsvdraws`, `voltimeplot`

`cortimeplot` *Plot correlations over time.*

Description

`cortimeplot` draws correlations over time.

Usage

```
cortimeplot(x, series, these = seq_len(nrow(x$y)), type = "cor",
  statistic = "mean", coldist = 1)

covtimeplot(x, series, these = seq_len(nrow(x$y)), type = "cov",
  statistic = "mean", coldist = 1)
```

Arguments

`x`	Object of class `'fsvdraws'`, usually resulting from a call to `fsvsample`.
`series`	Single number, coercible to integer. Indicates the series relative to which correlations are drawn.
`these`	Index vector containing the time points to plot. Defaults to `seq_len(nrow(x$y))`.
`type`	What to plot, usually "cor" or "cov".
`statistic`	Which posterior summary should be plotted, usually "mean".
`coldist`	Indicates how "different" adjacent colors should be.

Details

This function displays one component series' time-varying correlations with the other components series. Throws an error if correlations haven't been stored during sampling.

Value

Returns `x` invisibly.

See Also

Other plotting: `comtimeplot`, `corimageplot`, `corplot`, `facloadcredplot`, `facloaddensplot`, `facloadpairplot`, `facloadpointplot`, `facloadtraceplot`, `logvartimeplot`, `paratraceplot`, `plot.fsvdraws`, `voltimeplot`

covelement *Extract "true" model-implied covariances of two series only*

Description

covelement extracts the model-implied (time-varying) covariances between (exactly) two component series.

Usage

```
covelement(x, i, j, these = seq_len(nrow(x$y)))
```

Arguments

x	Object of class 'fsvsim', usually resulting from a call of the function fsvsim.
i	Index of component series 1.
j	Index of component series 2.
these	Vector indicating which points in time should be extracted, defaults to all.

Value

Vector with the requested covariances.

See Also

Other simulation: corelement, covmat.fsvsim

covmat *Generic extraction of covariance matrix*

Description

Generic function for extracting model-implied covariance matrices, either from the MCMC output, or from the simulated model.

Usage

```
covmat(x, ...)
```

Arguments

x	An object of class fsvdraws or fsvsim.
...	Arguments to be passed to methods.

Value

Structure containing the model-implied covariance matrix.

See Also

covmat.fsvsample covmat.fsvsim

`covmat.fsvdraws` *Extract posterior draws of the model-implied covariance matrix*

Description

`covmat` extracts draws from the model-implied covariance matrix from an `fsvdraws` object for all points in time which have been stored.

Usage

```
## S3 method for class 'fsvdraws'
covmat(x, ...)
```

Arguments

`x`	Object of class `'fsvdraws'`, usually resulting from a call of `fsvsample`.
`...`	Ignored.

Value

Array of dimension `m` times `m` times `draws` times `timepoints` containing the posterior draws for the model-implied covariance matrix.

Note

Currently crudely implemented as a double loop in pure R, may be slow.

See Also

covmat covmat.fsvsim

Other extractors: `runningcormat`, `runningcovmat`

Examples

```
## Not run:
set.seed(1)
sim <- fsvsim(n = 500, series = 3, factors = 1) # simulate
res <- fsvsample(sim$y, factors = 1, keeptime = "all") # estimate
covs <- covmat(res) # extract

# Trace plot of determinant of posterior covariance matrix
# at time t = n = 500:
detdraws <- apply(covs[,,,500], 3, det)
ts.plot(detdraws)
abline(h = mean(detdraws), col = 2)          # posterior mean
abline(h = median(detdraws), col = 4)        # posterior median
abline(h = det(covmat(sim)[,,500]), col = 3) # implied by DGP

# Trace plot of draws from posterior covariance of Sim1 and Sim2 at
# time t = n = 500:
ts.plot(covs[1,2,,500])
abline(h = covmat(sim)[1,2,500], col = 3) # "true" value

# Smoothed kernel density estimate:
plot(density(covs[1,2,,500], adjust = 2))

# Summary statistics:
summary(covs[1,2,,500])

## End(Not run)
```

`covmat.fsvsim` *Extract "true" model-implied covariance matrix for several points in time*

Description

`covmat` extracts the model-implied (time-varying) covariance matrix from an `fsvsim` object.

Usage

```
## S3 method for class 'fsvsim'
covmat(x, these = seq_len(nrow(x$y)), ...)
```

Arguments

`x`	Object of class `'fsvsim'`, usually resulting from a call of the function `fsvsim`.
`these`	Vector indicating which points in time should be extracted, defaults to all.
`...`	Ignored.

Value

Array of dimension `m` times `m` times `length(these)`, containing the model-implied covariance matrix.

Note

Currently crudely implemented as an R loop over all time points, may be slow.

See Also

covmat covmat.fsvsample

Other simulation: `corelement`, `covelement`

`expweightcov` *Computes the empirical exponentially weighted covariance matrix*

Description

A common way to get estimates for time-varying covariance matrices is the compute the exponentially weighted empirical covariance matrix.

Usage

```
expweightcov(dat, alpha = 4/126, hist = 180)
```

Arguments

`dat`	Matrix containing the data, with `n` rows (points in time) and `m` columns (component series).
`alpha`	Speed of decay.
`hist`	How far to go back in time?

Value

A `m` times `m` covariance matrix estimate.

facloadcredplot *Displays bivariate marginal posterior distribution of factor loadings.*

Description

facloadcredplot illustrates the bivariate marginals of the factor loadings distribution. It is a monochrome variant of facloadpairplot.

Usage

```
facloadcredplot(x, quants = c(0.01, 0.99))
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
quants	Posterior quantiles to be plotted.

Value

Returns x invisibly.

See Also

Other plotting: comtimeplot, corimageplot, corplot, cortimeplot, facloaddensplot, facloadpairplot, facloadpointplot, facloadtraceplot, logvartimeplot, paratraceplot, plot.fsvdraws, voltimeplot

facloaddensplot *Density plots of factor loadings draws*

Description

facloaddensplot draws kernel smoothed density plots of the marginal factor loadings posterior.

Usage

```
facloaddensplot(x, fsvsimobj = NULL, rows = 5, thesecols = NULL,
  xlim = NULL)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
fsvsimobj	To indicate data generating values in case of simulated data, pass an object of type fsvsim (usually the result of a call to fsvsim).
rows	Number of rows per page.
thesecols	Which factor loadings columns should be plotted? Defaults to 1:r.
xlim	Vector of length two containing lower and upper bounds of the horizontal axis. If NULL, these are automatically determined.

Value

Returns x invisibly.

See Also

Other plotting: comtimeplot, corimageplot, corplot, cortimeplot, facloadcredplot, facloadpairplot, facloadpointplot, facloadtraceplot, logvartimeplot, paratraceplot, plot.fsvdraws, voltimeplot

---

facloadpairplot *Displays bivariate marginal posterior distributions of factor loadings.*

Description

facloadpairplot illustrates the bivariate marginals of the factor loadings distribution. For a monochrome variant, see facloadcredplot.

Usage

```
facloadpairplot(x, maxpoints = 500, alpha = 20/maxpoints, cex = 3)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
maxpoints	The maximum amount of posterior draws to plot. If the number of draws stored in x exceeds this number, draws are thinned accordingly.
alpha	Level of transparency.
cex	Controls the size of the dots.

Value

Returns x invisibly.

See Also

Other plotting: comtimeplot, corimageplot, corplot, cortimeplot, facloadcredplot, facloaddensplot, facloadpointplot, facloadtraceplot, logvartimeplot, paratraceplot, plot.fsvdraws, voltimeplot

facloadpointplot *Displays point estimates of the factor loadings posterior.*

Description

facloadpointplot illustrates point estimates (mean, median, ...) of the estimated factor loadings matrix.

Usage

```
facloadpointplot(x, fsvsimobj = NULL, statistic = "median", cex = 6.5,
  alpha = 0.2, allpairs = FALSE, col = NULL)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
fsvsimobj	To indicate data generating values in case of simulated data, pass an object of type fsvsim (usually the result of a call to fsvsim).
statistic	Character string indicating which posterior statistic should be displayed.
cex	Controls the size of the dots.
alpha	Controls the level of transparency.
allpairs	Logical value; if set to TRUE, all possible pairwise combinations will be plotted.
col	Vector of length m (number of component series), containing rgb-type color codes used for plotting. Will be recycled if necessary.

Value

Returns x invisibly, throws a warning if there aren't any factors to plot.

See Also

Other plotting: comtimeplot, corimageplot, corplot, cortimeplot, facloadcredplot, facloaddensplot, facloadpairplot, facloadtraceplot, logvartimeplot, paratraceplot, plot.fsvdraws, voltimeplot

facloadtraceplot *Trace plots of factor loadings draws*

Description

facloadtraceplot draws trace plots of the factor loadings. Can be an important tool to check MCMC convergence if inference about (certain) factor loadings sought.

Usage

```
facloadtraceplot(x, fsvsimobj = NULL, thinning = NULL, maxrows = 10,
  ylim = NULL)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
fsvsimobj	To indicate data generating values in case of simulated data, pass an object of type fsvsim (usually the result of a call to fsvsim).
thinning	Plot every thinningth draw.
maxrows	Indicates the maximum number of rows to be drawn per page.
ylim	Vector of length two containing lower and upper bounds of the vertical axis. If NULL, these are automatically determined.

Value

Returns x invisibly.

See Also

Other plotting: comtimeplot, corimageplot, corplot, cortimeplot, facloadcredplot, facloaddensplot, facloadpairplot, facloadpointplot, logvartimeplot, paratraceplot, plot.fsvdraws, voltimeplot

---

findrestrict *Ad-hoc method for (weakly) identifying the factor loadings matrix*

Description

In factor SV models, the identification of the factor loadings matrix is often chosen through a preliminary static factor analysis. After a maximum likelihood factor model is fit to the data, variables are ordered as follows: The variable with the lowest loadings on all factors except the first (relative to it) is determined to lead the first factor, the variable with the lowest loadings on all factors except the first two (relative to these) is determined to lead the second factor, etc.

Usage

```
findrestrict(dat, factors, transload = abs, relto = "all")
```

Arguments

dat	Matrix containing the data, with n rows (points in time) and m columns (component series).
factors	Number of factors to be used.
transload	Function for transforming the estimated factor loadings before ordering. Defaults to the absolute value function.

relto — Can be 'none', 'current' or 'all'. If 'none', the series with the highest loadings is placed first, the series with the second highest is placed second, and so on. If 'current', the current factor loading is used as a reference, if 'all', all previous loadings are summed up to be the reference.

Value

A m times factors matrix indicating the restrictions.

Note

This function is automatically invoked by fsvsample if restrict is set to 'auto'.

See Also

ledermann

---

fsvsample — *Markov Chain Monte Carlo (MCMC) Sampling for the Factor Stochastic Volatility Model.*

Description

fsvsample simulates from the joint posterior distribution and returns the MCMC draws. It is the main workhorse to conduct inference for factor stochastic volatility models in this package.

Usage

```
fsvsample(y, factors = 1, draws = 1000, burnin = 1000, priormu = c(0,
  10), priorphiidi = c(10, 3), priorphifac = c(10, 3), priorsigmaidi = 1,
  priorsigmafac = 1, priorfacload = 1, priorng = c(1, 1),
  columnwise = FALSE, priorh0idi = "stationary",
  priorh0fac = "stationary", thin = 1, keeptime = "last",
  runningstore = 1, runningstorethin = 10, runningstoremoments = 2,
  quiet = FALSE, restrict = "auto", interweaving = 4,
  signswitch = FALSE, heteroskedastic = TRUE, priorhomoskedastic = NA,
  expert, startpara, startlatent, startlatent0, startfacload, startfac)
```

Arguments

y — Data matrix. Each of m rows is assumed to contain a single (univariate) series of length n.

factors — Number of latent factors to be estimated.

draws — Number of MCMC draws kept after burn-in.

burnin — Number of initial MCMC draws to be discarded.

priormu — Vector of length 2 denoting prior mean and standard deviation for unconditional levels of the idiosyncratic log-variance processes.

priorphiidi — Vector of length 2, indicating the shape parameters for the Beta prior distributions of the transformed parameters (phi+1)/2, where phi denotes the persistence of the idiosyncratic log-variances.

priorphifac — Vector of length 2, indicating the shape parameters for the Beta prior distributions of the transformed parameters (phi+1)/2, where phi denotes the persistence of the factor log-variances.

priorsigmaidi — Vector of length m containing the prior volatilities of log-variances. If priorsigmaidi has exactly one element, it will be recycled for all idiosyncratic log-variances.

priorsigmafac — Vector of length factors containing the prior volatilities of log-variances. If priorsigmafac has exactly one element, it will be recycled for all factor log-variances.

priorfacload — Either a matrix of dimensions m times factors with nonnegative elements or a single number. If a matrix is provided, its elements are the standard deviations of the Gaussian prior distributions for the factor loadings. If a single nonnegative number is provided, it will be recycled accordingly. If a single negative number is provided, a Normal-Gamma shrinkage prior for the factor loadings is assumed, where -priorfacload is interpreted as the shrinkage parameter a.

priorng — Two-element vector with positive entries indicating the Gamma hyperprior's hyperhyperparameters c and d.

columnwise — Set to TRUE if you want to use column-wise shrinkage and to FALSE for row-wise shrinkage. For details please see the paper by Kastner et al. (2016).

priorh0idi — Vector of length 1 or m, containing information about the Gaussian prior for the initial idiosyncratic log-variances. If an element of priorh0idi is a nonnegative number, the conditional prior of the corresponding initial log-variance h0 is assumed to be Gaussian with mean 0 and standard deviation priorh0idi times $sigma$. If an element of priorh0idi is the string 'stationary', the prior of the corresponding initial log volatility is taken to be from the stationary distribution, i.e. h0 is assumed to be Gaussian with mean 0 and variance $sigma^2/(1-phi^2)$.

priorh0fac — Vector of length 1 or factors, containing information about the Gaussian prior for the initial factor log-variances. If an element of priorh0fac is a nonnegative number, the conditional prior of the corresponding initial log-variance h0 is assumed to be Gaussian with mean 0 and standard deviation priorh0fac times $sigma$. If an element of priorh0fac is the string 'stationary', the prior of the corresponding initial log volatility is taken to be from the stationary distribution, i.e. h0 is assumed to be Gaussian with mean 0 and variance $sigma^2/(1-phi^2)$.

thin — Single number greater or equal to 1, coercible to integer. Every thinth MCMC draw is kept and returned. The default value is 1, corresponding to no thinning of the draws, i.e. every draw is stored.

keeptime — Either a number coercible to a positive integer, or a string equal to "all" or "last". If a number different from 1 is provided, only every keeptimeth latent log-volatility is being monitored. If, e.g., keeptime = 3, draws for the latent log-variances h_1,h_4,h_7,... will be kept. If keeptime is set to "all", this is equivalent to setting it to 1. If keeptime is set to "last" (the default), only draws for the very last latent log-variances h_n are kept.

runningstore Because most machines these days do not have enough memory to store all draws for all points in time, setting runningstore to an integer greater than 0 will cause fsvsample to store the first runningstoremoments ergodic moments of certain variables of interest. More specifically, mean, variance, skewness, etc. will be stored for certain variables if runningstore is set to a value...

- >= 1: Latent log-variances h_1,h_2,...,h_(n+r).
- >= 2: Latent factors f_1,...,f_r.
- >= 3: Latent volatilities sqrt(exp(h_1,h_2,...,h_(n+r))).
- >= 4: Conditional covariance matrix and the square roots of its diagonal elements.
- >= 5: Conditional correlation matrix.
- >= 6: Communalities, i.e. proportions of variances explained through the common factors.

runningstorethin

How often should the calculation of running moments be conducted? Set to a value > 1 if you want to avoid time consuming calculations at every MCMC iteration.

runningstoremoments

Selects how many running moments (up to 4) should be calculated.

quiet Logical value indicating whether the progress bar and other informative output during sampling should be omitted. The default value is FALSE, implying verbose output.

restrict Either "upper", "none", or "auto", indicating whether the factor loadings matrix should be restricted to have zeros above the diagonal ("upper"), whether all elements should be estimated from the data ("none"), or whether the function findrestrict should be invoked for a priori finding suitable zeros. Setting restrict to "upper" or "auto" often stabilizes MCMC estimation and can be important for identifying the factor loadings matrix, however, it generally is a strong prior assumption. Setting restrict to "none" is usually the preferred option if identification of the factor loadings matrix is of less concern but covariance estimation or prediction is the goal. Alternatively, restrict can be a logical matrix of dimension c(m, r) indicating which elements should be unrestricted (where restrict is FALSE) or zero (where restrict is TRUE).

interweaving The following values for interweaving the factor loadings are accepted:

- 0: No interweaving.
- 1: Shallow interweaving through the diagonal entries.
- 2: Deep interweaving through the diagonal entries.
- 3: Shallow interweaving through the largest absolute entries in each column.
- 4: Deep interweaving through the largest absolute entries in each column.

For details please see Kastner et al. (2016). A value of 4 is the highly recommended default.

signswitch Set to TRUE to turn on a random sign switch of factors and loadings. Note that the signs of each factor loadings matrix column and the corresponding factor cannot be identified from the likelihood.

heteroskedastic

Vector of length 1, 2, or `m + factors`, containing logical values indicating whether time-varying (`heteroskedastic = TRUE`) or constant (`heteroskedastic = FALSE`) variance should be estimated. If `heteroskedastic` is of length 2 it will be recycled accordingly, whereby the first element is used for all idiosyncratic variances and the second element is used for all factor variances.

priorhomoskedastic

Only used if at least one element of `heteroskedastic` is set to `FALSE`. In that case, `priorhomoskedastic` must be must be a matrix with positive entries and dimension c(factors + m, 2). Values in column 1 will be interpreted as shape and values in column 2 will be interpreted as the rate parameter of the corresponding inverse gamma prior distribution.

expert

optional named list of expert parameters for the univariate SV models (will be passed to the `stochvol` package). For most applications, the default values probably work best. Interested users are referred to Kastner and Frühwirth-Schnatter (2014) and Kastner (2016). If `expert` is provided, it may contain the following named elements:

- `parameterization`: Character string equal to `"centered"`, `"noncentered"`, `"GIS_C"`, or `"GIS_NC"`. Defaults to `"GIS_C"`.
- `mhcontrol`: Single numeric value controlling the proposal density of a Metropolis-Hastings (MH) update step when sampling `sigma`. If `mhcontrol` is smaller than 0, an independence proposal will be used, while values greater than zero control the stepsize of a log-random-walk proposal. Defaults to `-1`.
- `gammaprior`: Single logical value indicating whether a Gamma prior for `sigma^2` should be used. If set to `FALSE`, an Inverse Gamma prior is employed. Defaults to `TRUE`.
- `truncnormal`: Single logical value indicating whether a truncated Gaussian distribution should be used as proposal for draws of `phi`. If set to `FALSE`, a regular Gaussian prior is employed and the draw is immediately discarded when values outside the unit ball happen to be drawn. Defaults to `FALSE`.
- `mhsteps`: Either `1`, `2`, or `3`. Indicates the number of blocks used for drawing from the posterior of the parameters. Defaults to `2`.
- `proposalvar4sigmaphi`: Single positive number indicating the conditional prior variance of `sigma*phi` in the ridge *proposal* density for sampling `(mu, phi)`. Defaults to `10^8`.
- `proposalvar4sigmatheta`: Single positive number indicating the conditional prior variance of `sigma*theta` in the ridge *proposal* density for sampling `(mu, phi)`. Defaults to `10^12`.

startpara

optional named list, containing the starting values for the parameter draws. If supplied, `startpara` must contain three elements named `mu` (vector of length `m`), `phi` (vector of length `m+r`) and `sigma` (vector of length `m+r`). The elements of `mu` can be arbitrary numerical values, the `phi`s must be real numbers between `-1` and `1`, and the `sigma`s must be positive real numbers.

startlatent

optional numeric matrix of dimension `c(n, m + factors)`, containing the starting values of the latent log-variances.

`startlatent0` *optional* numeric vector of length `m + factors`, containing the starting values of the initial latent log-variances.

`startfacload` *optional* numeric matrix of dimension `c(m, factors)`, containing the starting values of the factor loadings.

`startfac` *optional* numeric matrix of dimension `c(factors, n)`, containing the starting values of the latent factors.

Details

For details concerning the factor SV algorithm please see Kastner et al. (2016), details about the univariate SV estimation can be found in Kastner and Frühwirth-Schnatter (2014).

Value

The value returned is a list object of class `fsvdraws` holding

- fArray containing factor draws from the posterior distribution.
- paraArray containing parameter draws form the posterior distribution.
- hArray containing idiosyncratic and factor log-variance draws.
- facloadArray containing draws from the posterior distribution of the factor loadings matrix.
- yMatrix containing the data supplied.
- runningstoreList whose elements contain ergodic moments of certain variables of interest. See argument `runningstore` for details about what is being stored here.

To display the output, use `print`, `summary` and `plot`. The `print` method simply prints the posterior draws (which is very likely a lot of output); the `summary` method displays the summary statistics currently stored in the object; the `plot` method `plot.fsvdraws` gives a graphical overview.

References

Kastner, G., Frühwirth-Schnatter, S., and Lopes, H.F. (2016). Efficient Bayesian Inference for Multivariate Factor Stochastic Volatility Models *Research Report Series / Department of Statistics and Mathematics*, **128**. https://epub.wu.ac.at/4875/.

Kastner, G. (2016). Dealing with stochastic volatility in time series using the R package stochvol. *Journal of Statistical Software*, **69**(5), 1–30, http://dx.doi.org/10.18637/jss.v069.i05.

Kastner, G. and Frühwirth-Schnatter, S. (2014). Ancillarity-sufficiency interweaving strategy (ASIS) for boosting MCMC estimation of stochastic volatility models. *Computational Statistics & Data Analysis*, **76**, 408–423, http://dx.doi.org/10.1016/j.csda.2013.01.002.

Examples

```
## Not run:
# Load exchange rate data (ships with stochvol):
data(exrates, package = "stochvol")
exrates$date <- NULL

# Compute the de-meaned percentage log returns:
dat <- 100 * logret(exrates, demean = TRUE)
```

```
# We are going to fit a one-factor model so the ordering is irrelevant
# NOTE that these are very few draws, you probably want more...
res <- fsvsample(dat, factors = 1, draws = 2000, burnin = 1000, runningstore = 6)

voltimeplot(res)

corimageplot(res, nrow(dat), plotCI = 'circle')

oldpar <- par(ask = TRUE)
plot(res)
par(oldpar)

## End(Not run)
```

`fsvsim` *Simulate data from a factor SV model*

Description

`fsvsim` generates simulated data from a factor SV model.

Usage

```
fsvsim(n = 1000, series = 10, factors = 1, facload = "dense", idipara,
  facpara, heteroskedastic = rep(TRUE, series + factors), df = Inf)
```

Arguments

`n`	Length of the series to be generated.
`series`	Number of component series `m`.
`factors`	Number of factors `r`.
`facload`	Can either be a matrix of dimension `m` times `r` or one of the keywords "dense" and "sparse". If "dense" is chosen, a (rather) dense lower triangular factor loadings matrix is randomly generated. If "sparse" is chosen, a (rather) sparse lower triangular factor loadings matrix is randomly generated.
`idipara`	*Optional* matrix of idiosyncratic SV parameters to be used for simulation. Must have exactly three columns containing the values of `mu`, `phi` and `sigma` for each of `m` series, respectively. If omitted, plausible values are generated.
`facpara`	*Optional* matrix of idiosyncratic SV parameters to be used for simulation. Must have exactly two columns containing the values of `phi` and `sigma` for each of `r` factors, respectively. If omitted, plausible values are generated.
`heteroskedastic`	Logical vector of length `m+r`. When `TRUE`, time-varying volatilities are generated; when `FALSE`, constant volatilities (equal to `mu`) are generated.
`df`	If not equal to Inf, the factors are misspecified (come from a t distribution instead of a Gaussian). Only used for testing.

Value

The value returned is a list object of class `fsvsim` holding

- yThe simulated data, stored in a `n` times `m` matrix with colnames 'Sim1', 'Sim2', etc.
- fThe simulated factors, stored in a `r` times `r` matrix.
- facloadFactor loadings matrix.
- facvolLatent factor log-variances for times 1 to `n`.
- facvol0Initial factor log-variances for time 0.
- facparaThe parameters of the factor volatility processes.
- idivolLatent idiosyncratic log-variances for times 1 to `n`.
- idivol0Initial idiosyncratic log-variances for time 0.
- idiparaThe parameters of the idiosyncratic volatility processes.

Note

This object can be passed to many plotting functions to indicate the data generating processes when visualizing results.

`ledermann` *Ledermann bound for the number of factors*

Description

In the static factor case, the Ledermann bound is the largest integer rank for which a unique decomposition of the covariance matrix is possible. (This is the largest possible number of factors which can be used for `factanal`.

Usage

```
ledermann(m)
```

Arguments

`m`	Number of component series.

Value

The Ledermann bound, a nonnegative integer.

See Also

preorder

`logret` *Computes the log returns of a vector-valued time series*

Description

`logret` computes the log returns of a multivariate time series, with optional de-meaning.

Usage

```
logret(dat, demean = FALSE, standardize = FALSE, ...)

## S3 method for class 'matrix'
logret(dat, demean = FALSE, standardize = FALSE, ...)

## S3 method for class 'data.frame'
logret(dat, ...)
```

Arguments

`dat`	The raw data, a matrix with `n` (number of timepoints) rows and `m` (number of component series) columns.
`demean`	Logical value indicating whether the data should be de-meaned.
`standardize`	Logical value indicating whether the data should be standardized (in the sense that each component series has an empirical variance equal to one).
`...`	Ignored.

Value

Matrix containing the log returns of the (de-meaned) data.

Methods (by class)

- `matrix`: (De-meaned) log returns
- `data.frame`: (De-meaned) log returns

See Also

fsvsample logret.matrix logret.data.frame

logvartimeplot *Plot log-variances over time.*

Description

logvartimeplot plots the idiosyncratic and factor log-variances over time.

Usage

```
logvartimeplot(x, fsvsimobj = NULL, show = "both", maxrows = 5)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
fsvsimobj	To indicate data generating values in case of simulated data, pass an object of type fsvsim (usually the result of a call to fsvsim).
show	If set to "fac", only factor log-volatilities will be displayed. If set to "idi", only idiosyncratic log-volatilities will be displayed. If set to "both", factor log-volatilities will be drawn first, followed by the idiosyncratic log-volatilities.
maxrows	Indicates the maximum number of rows to be drawn per page.

Details

This function displays the posterior distribution (mean +/- 2sd) of log-variances of both the factors and the idiosyncratic series. If these haven't been stored during sampling, logvartimeplot produces an error.

Value

Returns x invisibly.

See Also

Other plotting: comtimeplot, corimageplot, corplot, cortimeplot, facloadcredplot, facloaddensplot, facloadpairplot, facloadpointplot, facloadtraceplot, paratraceplot, plot.fsvdraws, voltimeplot

`orderident` *A posteriori factor order identification*

Description

`orderident` provides some (very ad-hoc) methods for identifying the ordering of the factors after running the (unrestricted) MCMC sampler by ordering according to the argument `method`.

Usage

```
orderident(x, method = "summed")
```

Arguments

`x`	Object of class `'fsvdraws'`, usually resulting from a call to `fsvsample`.
`method`	Methods currently supported:

- `summean` Sort by sum of mean loadings (descending).
- `summeaninv` Sort by sum of mean loadings (ascending).
- `summeanabs` Sort by sum of mean absolute loadings (descending).
- `summed` Sort by sum of median loadings (descending).
- `summedinv` Sort by sum of median loadings (ascending).
- `summedabs` Sort by sum of median absolute loadings (descending).
- `maxmed` Sort by maximum median loadings (descending).
- `maxmedinv` Sort by maximum median loadings (ascending).
- `maxmedrel` Sort by maximum median loadings, relative to the sum of all median loadings on that factor (descending).
- `maxmedabsrel` Sort by maximum absolute median loadings, relative to the sum of all median loadings on that factor (descending).

Value

Returns an object of class `'fsvdraws'` with adjusted ordering.

See Also

Other postprocessing: `signident`

paratraceplot *Trace plots of parameter draws*

Description

paratraceplot draws trace plots of all parameters (mu, phi,sigma). Can be an important tool to check MCMC convergence if inference about (certain) parameters is sought.

Usage

```
paratraceplot(x, fsvsimobj = NULL, thinning = NULL, maxrows = 3)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
fsvsimobj	To indicate data generating values in case of simulated data, pass an object of type fsvsim (usually the result of a call to fsvsim).
thinning	Plot every thinningth draw.
maxrows	Indicates the maximum number of rows to be drawn per page.

Value

Returns x invisibly.

See Also

Other plotting: comtimeplot, corimageplot, corplot, cortimeplot, facloadcredplot, facloaddensplot, facloadpairplot, facloadpointplot, facloadtraceplot, logvartimeplot, plot.fsvdraws, voltimeplot

plot.fsvdraws *Several factor SV plots*

Description

Draws a collection of plots to explore the posterior distribution of a fitted factor SV model.

Usage

```
## S3 method for class 'fsvdraws'
plot(x, fsvsimobj = NULL, ...)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
fsvsimobj	To indicate data generating values in case of simulated data, pass an object of type fsvsim (usually the result of a call to fsvsim).
...	Other arguments will be passed on to the subfunctions.

Value

Returns x invisibly.

See Also

Other plotting: comtimeplot, corimageplot, corplot, cortimeplot, facloadcredplot, facloaddensplot, facloadpairplot, facloadpointplot, facloadtraceplot, logvartimeplot, paratraceplot, voltimeplot

---

predcond *Predicts means and variances conditionally on the factors*

Description

predcond simulates from the posterior predictive distribution of the data, conditionally on realized values of the factors. This has the advantage that the predictive density can be written as the product of the marginals but introduces sampling uncertainty that grows with the number of factors used.

Usage

```
predcond(x, ahead = 1, each = 1, ...)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
ahead	Vector of timepoints, indicating how many steps to predict ahead.
each	Single integer (or coercible to such) indicating how often should be drawn from the posterior predictive distribution for each draw that has been stored during MCMC sampling.
...	Ignored.

Value

List of class fsvpredcond containing two elements:

- meansArray containing the draws of the predictive means.
- varsArray containing the draws of the predictive variances.

See Also

Other predictors: predcor, predcov, predh, predloglikWB, predloglik, predprecWB

Examples

```
## Not run:
set.seed(1)
sim <- fsvsim(n = 500, series = 4, factors = 1) # simulate
res <- fsvsample(sim$y, factors = 1) # estimate

# Predict 1 day ahead:
predobj <- predcond(res, each = 5)

# Draw from the predictive distribution:
preddraws <- matrix(rnorm(length(predobj$mean[,,1]),
                    mean = predobj$mean[,,1],
                    sd = predobj$vols[,,1]), nrow = 4)

# Visualize the predictive distribution
pairs(t(preddraws), col = rgb(0,0,0,.1), pch = 16)

## End(Not run)
```

---

predcor *Predicts correlation matrix*

Description

predcor simulates from the posterior predictive distribution of the model-implied correlation matrix.

Usage

```
predcor(x, ahead = 1, each = 1)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
ahead	Vector of timepoints, indicating how many steps to predict ahead.
each	Single integer (or coercible to such) indicating how often should be drawn from the posterior predictive distribution for each draw that has been stored during MCMC sampling.

Value

4-dimensional array containing draws from the predictive correlation distribution.

Note

Currently crudely implemented as a triple loop in pure R, may be slow.

See Also

Other predictors: predcond, predcov, predh, predloglikWB, predloglik, predprecWB

Examples

```
## Not run:
set.seed(1)
sim <- fsvsim(series = 3, factors = 1) # simulate
res <- fsvsample(sim$y, factors = 1) # estimate

# Predict 1, 10, and 100 days ahead:
predobj <- predcor(res, ahead = c(1, 10, 100))

# Trace plot of draws from posterior predictive distribution
# of the correlation of Sim1 and Sim2:
# (one, ten, and 100 days ahead):
plot.ts(predobj[1,2,,])

# Smoothed kernel density estimates of predicted covariance
# of Sim1 and Sim2:
plot(density(predobj[1,2,,"1"], adjust = 2))
lines(density(predobj[1,2,,"10"], adjust = 2), col = 2)
lines(density(predobj[1,2,,"100"], adjust = 2), col = 3)

## End(Not run)
```

---

predcov *Predicts covariance matrix*

Description

predcov simulates from the posterior predictive distribution of the model-implied covariance matrix.

Usage

```
predcov(x, ahead = 1, each = 1)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
ahead	Vector of timepoints, indicating how many steps to predict ahead.
each	Single integer (or coercible to such) indicating how often should be drawn from the posterior predictive distribution for each draw that has been stored during MCMC sampling.

Value

4-dimensional array containing draws from the predictive covariance distribution.

Note

Currently crudely implemented as a triple loop in pure R, may be slow.

See Also

Other predictors: predcond, predcor, predh, predloglikWB, predloglik, predprecWB

Examples

```
## Not run:
set.seed(1)
sim <- fsvsim(series = 3, factors = 1) # simulate
res <- fsvsample(sim$y, factors = 1) # estimate

# Predict 1, 10, and 100 days ahead:
predobj <- predcov(res, ahead = c(1, 10, 100))

# Trace plot of draws from posterior predictive distribution
# of the covariance of Sim1 and Sim2:
# (one, ten, and 100 days ahead):
plot.ts(predobj[1,2,,])

# Smoothed kernel density estimates of predicted covariance
# of Sim1 and Sim2:
plot(density(predobj[1,2,,"1"], adjust = 2))
lines(density(predobj[1,2,,"10"], adjust = 2), col = 2)
lines(density(predobj[1,2,,"100"], adjust = 2), col = 3)

## End(Not run)
```

---

predh *Predicts factor and idiosyncratic log-volatilities h*

Description

predh simulates from the posterior predictive distribution of the latent log-variances h, both for factors as well as for idiosyncratic series.

Usage

```
predh(x, ahead = 1, each = 1)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
ahead	Vector of timepoints, indicating how many steps to predict ahead.
each	Single integer (or coercible to such) indicating how often should be drawn from the posterior predictive distribution for each draw that has been stored during MCMC sampling.

Value

List of class fsvpredh containing two elements:

- idihArray containing the draws of the latent idiosyncratic log-volatilities.
- factorhArray containing the draws of the latent factor log-volatilities.

See Also

Other predictors: predcond, predcor, predcov, predloglikWB, predloglik, predprecWB

Examples

```
## Not run:
set.seed(1)
sim <- fsvsim(series = 3, factors = 1) # simulate
res <- fsvsample(sim$y, factors = 1) # estimate

# Predict 1, 10, and 100 days ahead:
predobj <- predh(res, ahead = c(1, 10, 100))

# Trace plot of draws from posterior predictive factor log-variance
# (one, ten, and 100 days ahead):
plot.ts(predobj$factorh[1,,])

# Smoothed kernel density estimates of predicted volas:
plot(density(exp(predobj$factorh[1,,"1"]/2), adjust = 2))
lines(density(exp(predobj$factorh[1,,"10"]/2), adjust = 2), col = 2)
lines(density(exp(predobj$factorh[1,,"100"]/2), adjust = 2), col = 3)

## End(Not run)
```

---

predloglik *Evaluates the predictive log likelihood using the predicted covariance matrix*

Description

predloglik approximates the predictive log likelihood by simulating from the predictive distribution of the covariance matrix and evaluating the corresponding multivariate normal distribution.

Usage

```
predloglik(x, y, ahead = 1, each = 1, alldraws = FALSE)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
y	Matrix of dimension length(ahead) times m where the predictive density should be evaluated.
ahead	Vector of timepoints, indicating how many steps to predict ahead.
each	Single integer (or coercible to such) indicating how often should be drawn from the posterior predictive distribution for each draw that has been stored during MCMC sampling.
alldraws	Should all the draws be returned or just the final results? (Can be useful to assess convergence.)

Value

Vector of length length(ahead) with log predictive likelihoods.

See Also

Uses predcov. If m is large but only few factors are used, consider also using predloglikWB.

Other predictors: predcond, predcor, predcov, predh, predloglikWB, predprecWB

Examples

```
## Not run:
set.seed(1)

# Simulate a time series of length 1100:
sim <- fsvsim(n = 1100, series = 3, factors = 1)
y <- sim$y

# Estimate using only 1000 days:
res <- fsvsample(y[seq_len(1000),], factors = 1)

# Evaluate the 1, 10, and 100 days ahead predictive log
# likelihood:
ahead <- c(1, 10, 100)
scores <- predloglik(res, y[1000+ahead,], ahead = ahead, each = 10)
print(scores)

## End(Not run)
```

predloglikWB *Evaluates the predictive log likelihood using the Woodbury identity*

Description

predloglikWB approximates the predictive log likelihood exploiting the factor structure and using the Woodbury idenitity and the corresponding matrix determinant lemma. This is recommended only if many series and few factors are present.

Usage

```
predloglikWB(x, y, ahead = 1, each = 1, alldraws = FALSE)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
y	Matrix of dimension length(ahead) times m where the predictive density should be evaluated.
ahead	Vector of timepoints, indicating how many steps to predict ahead.
each	Single integer (or coercible to such) indicating how often should be drawn from the posterior predictive distribution for each draw that has been stored during MCMC sampling.
alldraws	Should all the draws be returned or just the final results? (Can be useful to assess convergence.)

Value

Vector of length length(ahead) with log predictive likelihoods.

Note

Currently crudely implemented as a triple loop in pure R, may be slow.

See Also

Uses predprecWB. If m is small or many factors are used, consider also using predcov.

Other predictors: predcond, predcor, predcov, predh, predloglik, predprecWB

Examples

```
## Not run:
set.seed(1)

# Simulate a time series of length 1100:
sim <- fsvsim(n = 1100, series = 3, factors = 1)
y <- sim$y
```

```
# Estimate using only 1000 days:
res <- fsvsample(y[seq_len(1000),], factors = 1)

# Evaluate the 1, 10, and 100 days ahead predictive log
# likelihood:
ahead <- c(1, 10, 100)
scores <- predloglikWB(res, y[1000+ahead,], ahead = ahead, each = 10)
print(scores)

## End(Not run)
```

---

predprecWB *Predicts precision matrix and its determinant (Woodbury variant)*

Description

predprecWB simulates from the posterior predictive distribution of the model-implied precision matrix and its determinant using the Woodbury matrix identity and the matrix determinant lemma

Usage

```
predprecWB(x, ahead = 1, each = 1)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
ahead	Vector of timepoints, indicating how many steps to predict ahead.
each	Single integer (or coercible to such) indicating how often should be drawn from the posterior predictive distribution for each draw that has been stored during MCMC sampling.

Value

List containing two elements:

- precisionArray containing the draws of the predicted precision matrix.
- precisionlogdetMatrix containing the draws of the determinant of the predicted precision matrix.

Note

Currently crudely implemented as a triple loop in pure R, may be slow.

See Also

Usually used for evaluating the predictive likelihood when many series but few factors are used, see predloglik and predloglikWB.

Other predictors: predcond, predcor, predcov, predh, predloglikWB, predloglik

`preorder` *Ad-hoc methods for determining the order of variables*

Description

In factor SV models, the ordering of variables is often chosen through a preliminary static factor analysis. These methods are implemented in `preorder`. After a maximum likelihood factor model fit to the data, factor loadings are ordered as follows: The variable with the highest loading on factor 1 is placed first, the variable with the highest loading on factor 2 second (unless this variable is already placed first, in which case the variable with the second highest loading is taken).

Usage

```
preorder(dat, factors = ledermann(ncol(dat)), type = "fixed",
  transload = identity)
```

Arguments

`dat`	Matrix containing the data, with `n` rows (points in time) and `m` columns (component series).
`factors`	Number of factors to be used, defaults to the Ledermann bound.
`type`	Can be "fixed" or "dynamic". The option "fixed" means that that a `factors`-factor model is fit once and the entire ordering is determined according to this fit (the default). The option "dynamic" means that the model is re-fit `factors` times with the number of factors going from 1 to `factors` and in each round the correspondingly largest loading is chosen.
`transload`	Function for transforming the estimated factor loadings before ordering. Defaults to the identity function.

Value

A vector of length `m` with the ordering found.

See Also

ledermann

print.fsvdraws *Pretty printing of an fsvsdraws object*

Description

Pretty printing of an fsvsdraws object

Usage

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

Arguments

x	Object of class 'fsvdraws', usually resulting from a call of fsvsample.
...	Ignored.

Value

Returns x invisibly.

runningcormat *Extract summary statistics for the posterior correlation matrix which have been stored during sampling*

Description

runningcormat extracts summary statistics from the model-implied correlation matrix from an fsvdraws object for one point in time.

Usage

```
runningcormat(x, i, statistic = "mean", type = "cor")
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call of fsvsample.
i	A single point in time.
statistic	Indicates which statistic should be extracted. Defaults to 'mean'.
type	Indicates whether covariance (cov) or correlation (cor) should be extracted.

Value

Matrix containing the requested correlation matrix summary statistic.

See Also

Other extractors: covmat.fsvdraws, runningcovmat

Examples

```
## Not run:
set.seed(1)
sim <- fsvsim(n = 500, series = 3, factors = 1) # simulate
res <- fsvsample(sim$y, factors = 1, runningstore = 6) # estimate

cor100mean <- runningcormat(res, 100) # extract mean at t = 100
cor100sd <- runningcormat(res, 100, statistic = "sd") # extract sd
lower <- cor100mean - 2*cor100sd
upper <- cor100mean + 2*cor100sd

true <- cov2cor(covmat(sim, 100)[,,1]) # true value

# Visualize mean +/- 2sd and data generating values
par(mfrow = c(3,3), mar = c(2, 2, 2, 2))
for (i in 1:3) {
 for (j in 1:3) {
  plot(cor100mean[i,j], ylim = range(lower, upper), pch = 3,
  main = paste(i, j, sep = ' vs. '), xlab = '', ylab = '')
  lines(c(1,1), c(lower[i,j], upper[i,j]))
  points(true[i,j], col = 3, cex = 2)
 }
}

## End(Not run)
```

`runningcovmat` *Extract summary statistics for the posterior covariance matrix which have been stored during sampling*

Description

`runningcovmat` extracts summary statistics from the model-implied covariance matrix from an `fsvdraws` object for one point in time.

Usage

```
runningcovmat(x, i, statistic = "mean", type = "cov")
```

Arguments

`x`	Object of class `'fsvdraws'`, usually resulting from a call of `fsvsample`.
`i`	A single point in time.
`statistic`	Indicates which statistic should be extracted. Defaults to `'mean'`.
`type`	Indicates whether covariance (`cov`) or correlation (`cor`) should be extracted.

Value

Matrix containing the requested covariance matrix summary statistic.

See Also

Other extractors: covmat.fsvdraws, runningcormat

Examples

```
## Not run:
set.seed(1)
sim <- fsvsim(n = 500, series = 3, factors = 1) # simulate
res <- fsvsample(sim$y, factors = 1, runningstore = 6) # estimate

cov100mean <- runningcovmat(res, 100) # extract mean at t = 100
cov100sd <- runningcovmat(res, 100, statistic = "sd") # extract sd
lower <- cov100mean - 2*cov100sd
upper <- cov100mean + 2*cov100sd

true <- covmat(sim, 100) # true value

# Visualize mean +/- 2sd and data generating values
par(mfrow = c(3,3), mar = c(2, 2, 2, 2))
for (i in 1:3) {
 for (j in 1:3) {
  plot(cov100mean[i,j], ylim = range(lower, upper), pch = 3,
  main = paste(i, j, sep = ' vs. '), xlab = '', ylab = '')
  lines(c(1,1), c(lower[i,j], upper[i,j]))
  points(true[i,j,1], col = 3, cex = 2)
 }
}

## End(Not run)
```

---

signident *A posteriori sign identification*

Description

signident provides methods for identifying the signs of the factor loadings after running the MCMC sampler

Usage

```
signident(x, method = "maximin", implementation = 3)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
method	Can be "diagonal" or "maximin". If "diagonal" is chosen, the diagonal elements of the factor loadings matrix are assumed to have positive signs and the others are arranged accordingly. If "maximin" is chosen, for each factor, signident looks for the series where the minimum absolute loadings are biggest and chooses this series to have positive loadings.
implementation	Either 1, 2, or 3 (the default). Determines how the reordering is implemented. Should not be necessary to depart from the default.

Value

Returns an object of class 'fsvdraws' with adjusted factors and factor loadings. Moreover, a list element called 'identifier' is added, providing the numbers of the series used for identification and the corresponding minimum distances to zero.

See Also

Other postprocessing: orderident

Examples

```
## Not run:
set.seed(1)
sim <- fsvsim(series = 8, factors = 2) # simulate
res <- fsvsample(sim$y, factors = 2, signswitch = TRUE,
                 draws = 2000, burnin = 1000) # estimate

# Plot unidentified loadings:
facloaddensplot(res, fsvsimobj = sim, rows = 8)

# Identify:
res <- signident(res)

# Plot identified loadings:
facloaddensplot(res, fsvsimobj = sim, rows = 8)

## End(Not run)
```

---

voltimeplot *Plot series-specific volatilities over time.*

Description

voltimeplot plots the marginal volatilities over time, i.e. the series-specific conditional standard deviations. If these haven't been stored during sampling (because runningstore has been set too low), voltimeplot throws a warning.

Usage

```
voltimeplot(x, these = seq_len(nrow(x$y)), legend = "topright", ...)
```

Arguments

x	Object of class 'fsvdraws', usually resulting from a call to fsvsample.
these	Index vector containing the time points to plot. Defaults to seq_len(nrow(x$y)), i.e., all timepoints.
legend	Where to position the link{legend}. If set to NULL, labels will be put directly next to the series. Defaults to "topright".
...	Additional parameters will be passed on to ts.plot.

Value

Returns x invisibly.

See Also

Other plotting: comtimeplot, corimageplot, corplot, cortimeplot, facloadcredplot, facloaddensplot, facloadpairplot, facloadpointplot, facloadtraceplot, logvartimeplot, paratraceplot, plot.fsvdraws

Index