Package 'dse2'

July 11, 2007

Title Dynamic Systems Estimation - extensions

Description Multivariate Time Series - extensions.See ?00dse-Intro for more details.

Depends R (>= 2.5.0), setRNG, tframe (>= 2007.5-3), dse1 (>= 2007.7-1)

Version 2007.7-1

Date 2007-07-11

LazyLoad yes

License Free. See the LICENCE file for details.

Author Paul Gilbert <pgilbert@bank-banque-canada.ca>

Maintainer Paul Gilbert <pgilbert@bank-banque-canada.ca>

URL http://www.bank-banque-canada.ca/pgilbert

R topics documented:

`coef.TSmodelEstEval`

Specific Methods for coef

Description

See the generic function description.

Usage

```
## S3 method for class 'TSmodelEstEval':
coef(object, criterion.args=NULL, ...)
## S3 method for class 'TSestModelEstEval':
coef(object, criterion.args=NULL, ...)
```

Arguments

`object`	an object (model) from which to extract coefficients(parameters).
`criterion.args`	arguments to be passed to this method when it is called by `EstEval`.
`...`	(further arguments, currently disregarded).

Details

The methods ***.ee are intended mainly to be called from `EstEval` as criterion for evaluating an estimation method. See `coef`.

See Also

`EstEval` `coef`

`combine.forecastCov`

Combine 2 Forecast Cov Objects

Description

Combine 2 forecastCov type objects.

Usage

```
## S3 method for class 'forecastCov':
combine(e1, e2)
## S3 method for class 'forecastCovEstimatorsWRTdata':
combine(e1, e2)
## S3 method for class 'forecastCovEstimatorsWRTtrue':
combine(e1, e2)
```

Arguments

`e1, e2`	Objects as returned by functions which calculate forecast covariances.

Details

Functions which calculate forecast covariances return lists. Usually multiple estimation techniques or models will be combined together when the object is first formed. However, it is sometimes useful to add results calculated later without re-doing the initial object.

Value

An object as returned by functions which calculate forecast covariances.

See Also

`combine`, `forecastCovEstimatorsWRTdata`, `forecastCovEstimatorsWRTtrue` `forecastCov`

Examples

```
#z <-  combine(obj1, obj2)
```

`distribution.MonteCarloSimulations`
Generate distribution plots of Monte Carlo simulations

Description

Generate distribution plots of Monte Carlo simulations.

Usage

```
## S3 method for class 'MonteCarloSimulations':
distribution(obj,
    series=seq(dim(obj$simulations)[2]),
    x.sections=TRUE, periods=1:3, graphs.per.page=5, ...)
```

Arguments

`obj`	The result of MonteCarloSimulations.
`series`	The series which should be plotted. The default gives all series.
`x.sections`	If TRUE then kernel density estimates are plotted for periods indicated by periods. If FALSE then a time series plots of the mean and estimates 1 and 2 standard deviations from the mean. Periods is ignored if x.sections is FALSE.
`periods`	The periods at which the distribution should be calculated and plotted. The default gives the first three.
`graphs.per.page`	integer indicating number of graphs to place on a page.
`...`	(further arguments, currently disregarded).

Details

Kernel estimates of the densities (series by series, not joint densities) are estimated using ksmooth (if available) or density (if available) to produces density plots. Output graphics can be paused between pages by setting par(ask=TRUE).

Value

None

See Also

`tfplot.MonteCarloSimulations`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(eg1.DSE.data.diff)
z <-  MonteCarloSimulations(model)
distribution(z)
```

`distribution` *Plot distribution of estimates*

Description

Plot distribution of estimates.

Usage

```
distribution(obj, ...)
 ## S3 method for class 'TSdata':
 distribution(obj, ..., bandwidth=0.2,
         select.inputs = seq(length= nseriesInput(obj)),
         select.outputs= seq(length=nseriesOutput(obj)))
 ## Default S3 method:
 distribution(obj, ..., bandwidth=0.2, series=NULL)
 ## S3 method for class 'coefEstEval':
 distribution(obj, ...,  Sort=FALSE, bandwidth=0.2,
    graphs.per.page=5)
 ## S3 method for class 'rootsEstEval':
 distribution(obj, ..., mod=TRUE, invert=FALSE, Sort=FALSE,
    bandwidth=0.2, select=NULL)
```

Arguments

`obj`	an object as returned by EstEval.
`Sort`	if Sort is true then sort is applied. This helps (a bit) with estimation methods like black.box which may not return parameters of the same length or in the same order.
`bandwidth`	passed to `density` or `ksmooth`.
`graphs.per.page`	integer indicating number of graphs to place on a page.
`series`	series to be plotted. (passed to selectSeries)
`select.inputs`	series to be plotted. (passed to selectSeries)
`select.outputs`	series to be plotted. (passed to selectSeries)
`...`	other objects to be plotted (not working for some methods).
`invert`	logical indicating if the inverse of roots should be plotted
`mod`	logical indicating if the modulus of roots should be plotted
`select`	integer vector indicating roots to be plotted. If select is not NULL then roots are sorted by magnitude and only the indicated roots are plotted. For example, select=c(1,2) will plot only the two largest roots.

Details

`ksmooth` is applied if available to get a smoothed estimate of the distribution of the estimates. If `ksmooth` is not available then `density` is applied if it is available.

Value

None

See Also

EstEval

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(TSdata(output=outputData(eg1.DSE.data.diff)), max.lag=2)
# now use this as the true model
z <-  EstEval(model,
    estimation="estVARXls", estimation.args=list(max.lag=2))
distribution(z)
tfplot(z)
```

EstEval *Evaluate an estimation method*

Description

Evaluate an estimation method.

Usage

```
EstEval(model, replications=100, rng=NULL, quiet=FALSE,
                    simulation.args=NULL,
                    estimation=NULL, estimation.args=NULL,
                    criterion ="coef", criterion.args =NULL)
is.EstEval(obj)
```

Arguments

model	A TSmodel.
replications	The number of simulations.
rng	The RNG and starting seed.
quiet	If TRUE then no information is printed during estimation.
simulation.args	A list of any arguments to pass to simulate.
estimation	A character string indicating the estimation routine to use.
estimation.args	A list of any arguments to pass to the estimation routine.
criterion	A function to apply to the results of estimation to extract the information which is to be retained.
criterion.args	A list of any arguments to be passed to the criterion function.
obj	an object.

Details

estimation.args and criterion.args should be NULL if no args are needed. If model is an object of class 'EstEval' or 'simulation' then the model and the seed!!! are extracted so the evaluation will be based on the same generated sample. criterion can be 'coef', 'roots', 'TSmodel', 'TSestModel' With the default (coef) or with TSmodel the other criteria can be reconstructed (when the estimation method finds a known form for the model - which is not always the case, for example with estBlackBox methods). If criterion = 'roots' then criterion.args= list(verbose=FALSE) is advised.

Value

A list with element `result` of length replications, each element containing the results of criterion(estimation(simulate(model))). Other elements of the list contain information from the supplied arguments.

See Also

`simulate` `MonteCarloSimulations` `distribution` `forecastCovWRTtrue`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(TSdata(output=outputData(eg1.DSE.data.diff)))
z <-  EstEval(model,
   estimation="estVARXls", estimation.args=list(max.lag=2))
tfplot(z)
zz <-  EstEval(model,
   estimation="estVARXls", estimation.args=list(max.lag=2),
   simulation.args=list(sampleT=50, sd=1.5))
is.EstEval(z)
```

`estimateModels` *Estimate Models*

Description

Estimate models using given estimation method

Usage

```
estimateModels(data, estimation.sample = NULL, trend =FALSE, quiet =FALSE,
                     estimation.methods = NULL)
is.estimatedModels(obj)
```

Arguments

`data`	An object of class TSdata.
`estimation.methods`	A named list with the names indicating the estimation method and the value associated with the name is a list of arguments for each the method indicated. Its value should be NULL if no args are needed.

`estimation.sample`
: An integer indicating the number of points in the sample to use for estimation. If it is NULL the whole sample is used.

`trend`
: If trend is TRUE then a linear trend is calculated and returned as the element `trend.coef`.

`quiet`
: If quiet is TRUE then most printing and some warning messages are suppressed.

`obj`
: An object.

Details

Estimate models from data with estimation methods indicated by estimation.methods. This is primarily a utility for other functions.

Value

Element `multi.model` in the result is a list of the same length as estimation.methods with resulting models as elements.

See Also

`EstEval`, `outOfSample.forecastCovEstimatorsWRTdata`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
z <-  estimateModels(eg1.DSE.data.diff, estimation.methods= list(
        bft=list(verbose=FALSE),
        estVARXar=list(max.lag=3)))
```

---

estimatorsHorizonForecastsWRTdata

Estimate models and forecast at given horizons

Description

Estimate models and forecast at given horizons.

Usage

```
estimatorsHorizonForecastsWRTdata(data,
                    estimation.sample=.5, horizons=1:12,quiet=FALSE,
                    estimation.methods=NULL)
```

Arguments

`data`
: A TSdata object.

`estimation.methods`
: A list of estimation methods to use. (See estimateModels.)

`estimation.sample`
: The portion of the sample to use for estimation.

`horizons`
: The horizons for which forecasts are to be produced.

`quiet`
: If true no estimation information is printed.

Details

estimation.sample indicates the part of the data to use for estimation. If estimation.sample is less than or equal 1.0 it is used to indicate the portion of points to use for estimation. Otherwise it should be an integer and is used to indicate the number of points from the beginning of the sample to use for estimation.

Value

A list of forecasts at different horizons as returned by horizonForecasts.

See Also

`estimateModels`, `horizonForecasts`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
z <-  estimatorsHorizonForecastsWRTdata(eg1.DSE.data.diff,
    estimation.methods=list(estVARXls=list(max.lag=3),
                estVARXar=list(max.lag=3)))
```

---

excludeForecastCov *Filter Object to Remove Forecasts*

Description

Filter object to remove forecasts.

Usage

```
excludeForecastCov(obj, exclude.series=NULL)
```

Arguments

`obj`	An object as returned by stripMine.
`exclude.series`	An indication of series to which should be excluded.

Details

Exclude results which depend on the indicated series from a (forecastCovEstimatorsWRTdata.subsets forecastCov) object.

Value

The returned result is a forecastCov object like obj, but filtered to remove any forecasts from models which depend on the series which are indicated for exclusion.

See Also

`minForecastCov`, `selectForecastCov`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
z <- stripMine(eg1.DSE.data.diff, essential.data=c(1,2),
                    estimation.methods=list(estVARXls=list(max.lag=3)))
z <-  excludeForecastCov(z, exclude.series=3)
```

---

extractforecastCov *Extract Forecast Covariance*

Description

extract forecastCov from objects

Usage

```
extractforecastCov(e, n)
## S3 method for class 'forecastCovEstimatorsWRTdata':
extractforecastCov(e, n)
## S3 method for class 'forecastCovEstimatorsFromModel':
extractforecastCov(e, n)
```

Arguments

e	A "forecastCovEstimatorsWRTdata", "forecastCov" object.
n	A vector on integers.

Details

Select a subset of models and their forecast covariances from a larger object.

Value

A forecastCov object.

See Also

forecastCov

---

featherForecasts *Multiple Horizon-Step Ahead Forecasts*

Description

Calculate multiple horizon-step ahead forecasts.

Usage

```
featherForecasts(obj, ...)
## S3 method for class 'TSestModel':
featherForecasts(obj, data=NULL, ...)
## S3 method for class 'TSdata':
featherForecasts(obj, model, ...)
## S3 method for class 'TSmodel':
featherForecasts(obj, data, horizon=36,
         from.periods =NULL, ...)
is.featherForecasts(obj)
```

Arguments

`obj`	an object of class TSmodel.
`data`	an object of class TSdata.
`model`	an object of class TSmodel.
`from.periods`	the starting points to use for forecasts.
`horizon`	the number of periods to forecast.
`...`	for a TSmodel additional arguments are passed to `l()`

Details

Calculate multiple horizon-step ahead forecasts ie. use the samples indicated by from.periods to calculate forecasts for horizon periods. Thus, for example, the result of featherForecasts(model, data, from.periods=c(200,250,300)) would be forecasts for 1 through 36 steps ahead (the default), starting at the 200th,250th, and 300th point of outputData(data). This function assumes that inputData(data) (the exogenous variable) is as long as necessary for the most future forecast.

Value

The result is a list of class featherForecasts with elements `model` (a TSestModel), `data`, `from.periods`, `featherForecasts`. The element `featherForecasts` is a list with length(from.periods) elements, each of which is a tframed matrix. There is a plot method for this class.

See Also

`forecast`, `horizonForecasts`

Examples

```
data("egJofF.1dec93.data", package="dse1")
model <- estVARXls(egJofF.1dec93.data)
pr <- featherForecasts(model, egJofF.1dec93.data)
tfplot(pr)
```

`forecastCovCompiled` *Forecast covariance for different models - internal*

Description

See forecastCov.

Usage

```
forecastCovCompiled(model, data, horizons = 1:12,
     discard.before=minimumStartupLag(model))
## S3 method for class 'ARMA':
forecastCovCompiled(model, data, horizons = 1:12,
     discard.before=minimumStartupLag(model))
## S3 method for class 'SS':
forecastCovCompiled(model, data, horizons = 1:12,
     discard.before=minimumStartupLag(model))
## S3 method for class 'innov':
forecastCovCompiled(model, data, horizons = 1:12,
     discard.before=minimumStartupLag(model))
## S3 method for class 'nonInnov':
forecastCovCompiled(model, data, horizons = 1:12,
     discard.before=minimumStartupLag(model))
 forecastCovSingleModel( model, data=NULL, horizons=1:12,
      discard.before=minimumStartupLag(model), compiled=.DSEflags()$COMPILED
```

Arguments

`obj`	TSdata or one or more TSmodels or TSestModels
`data`	an object of class TSdata.
`discard.before`	period before which forecasts should be discarded when calculating covariance.
`horizons`	horizons for which forecast covariance should be calculated.
`zero`	if TRUE the covariance is calculated for a forecast of zero.
`trend`	if TRUE the covariance is calculated for a forecast of trend.
`estimation.sample`	portion of the sample to use for calculating the trend.
`compiled`	a logical indicating if compiled code should be used. (Usually true except for debugging.)
`...`	arguments passed to other methods.

Details

Not to be called by users. See `forecastCov`.

Value

A list with the forecast covariance for supplied models on the given sample. This is in the element `forecastCov` of the result. Other elements contain information in the arguments.

`forecastCovEstimatorsWRTdata`
Calculate Forecast Cov of Estimators WRT Data

Description

forecast covariance of estimated models with respect to a given sample

Usage

```
forecastCovEstimatorsWRTdata(data, estimation.sample=NULL,
                    compiled=.DSEflags()$COMPILED, discard.before=10,
                    horizons=1:12, zero=FALSE, trend=FALSE,quiet=FALSE,
                    estimation.methods=NULL)
is.forecastCovEstimatorsWRTdata(obj)
```

Arguments

`data`	an object of class TSdata.
`estimation.methods`	a list as used by estimateModels.
`discard.before`	an integer indicating the number of points in the beginning of forecasts to discard for calculating covariances.
`zero`	if TRUE then forecastCov is also calculated for a forecast of zero.
`trend`	if TRUE then forecastCov is also calculated for a forecast of a linear trend.
`estimation.sample`	an integer indicating the number of points in the sample to use for estimation. If it is NULL the whole sample is used.
`horizons`	horizons for which forecast covariance should be calculated.
`quiet`	if TRUE then estimation information is not printed.
`compiled`	a logical indicating if the compiled version of the code should be used. (FALSE would typically only be used for debugging.)
`obj`	an object.

Details

Calculate the forecasts cov of models estimated from data with estimation methods indicated by estimation.methods (see estimateModels). estimation.sample is an integer indicating the number of points in the sample to use for estimation. If it is NULL the whole sample is used.

Value

A list with the forecast covariance for supplied models on the given sample. This is in the element `forecastCov` of the result. Other elements contain information in the arguments.

See Also

`outOfSample.forecastCovEstimatorsWRTdata`, `estimateModels`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
z <- forecastCovEstimatorsWRTdata(eg1.DSE.data.diff,
    estimation.methods=list(estVARXls=list(max.lag=4)))
```

---

forecastCovEstimatorsWRTtrue

Compare Forecasts Cov Relative to True Model Output

Description

Usage

```
forecastCovEstimatorsWRTtrue(true.model, rng=NULL,
                    simulation.args=NULL,
                    est.replications = 2, pred.replications = 2,
                    discard.before = 10, horizons = 1:12, quiet =FALSE,
                    estimation.methods=NULL, compiled=.DSEflags()$COMPILED)
is.forecastCovEstimatorsWRTtrue(obj)
```

Arguments

`true.model`	An object of class TSmodel.
`estimation.methods`	A list as used by estimateModels.
`simulation.args`	an arguments to be passed to simulate.
`est.replications`	An arguments to be passed to simulate.
`pred.replications`	An arguments to be passed to simulate.
`discard.before`	An integer indicating the number of points in the beginning of forecasts to discard for calculating covariances.
`horizons`	Horizons for which forecast covariance should be calculated.
`rng`	If specified then it is used to set RNG.
`quiet`	If TRUE then some messages are not printed.
`compiled`	a logical indicating if the compiled version of the code should be used. (FALSE would typically only be used for debugging.)
`obj`	an object.

Details

Calculate the forecasts cov of models estimated from simulations of true.model with estimation methods indicated by estimation.methods (see estimateModels). This function makes multiple calls to forecastCovWRTtrue.

Value

The returned results has element `forecastCov.true, forecastCov.zero, forecastCov.trend` containing covariances averaged over estimation replications and simulation replications (forecasts will not change but simulated data will). `forecastCov` a list of the same length as estimation.methods with each element containing covariances averaged over estimation replications and simulation replications. `estimatedModels` a list of length est.replications, with each elements as returned by estimateModels, thus each element has `multi.model` as a subelement containing models for different estimation techniques. So, eg. `estimatedModels[[2]]$multi.model[[1]]` in the result will be the model from the first estimation technique in the second replication.

See Also

`forecastCovWRTtrue` `forecastCovEstimatorsWRTdata`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
true.model <- estVARXls(eg1.DSE.data.diff) # just to have a starting model
z <-  forecastCovEstimatorsWRTtrue(true.model,
    estimation.methods=list(estVARXls=list(max.lag=4)))
```

---

forecastCov *Forecast covariance for different models*

Description

Calculate the forecast covariance for different models.

Usage

```
is.forecastCov(obj)
forecastCov(obj, ..., data=NULL, horizons=1:12, discard.before=NULL,
   zero=FALSE, trend=FALSE, estimation.sample= NULL, compiled=.DSEflags()$CO
## S3 method for class 'TSmodel':
forecastCov(obj, ..., data=NULL,
   horizons=1:12, discard.before=NULL,
   zero=FALSE, trend=FALSE, estimation.sample= periods(data), compiled=.DSEf
## S3 method for class 'TSestModel':
forecastCov(obj, ..., data=TSdata(obj),
   horizons=1:12, discard.before=NULL, zero=FALSE, trend=FALSE,
   estimation.sample= periods(TSdata(obj)), compiled=.DSEflags()$COMPILED)
## S3 method for class 'TSdata':
forecastCov(obj, ..., data=NULL,
   horizons=1:12, discard.before=1,
   zero=FALSE, trend=FALSE, estimation.sample= NULL,
    compiled=.DSEflags()$COMPILED)
```

Arguments

`obj`	TSdata or one or more TSmodels or TSestModels
`data`	an object of class TSdata.
`discard.before`	period before which forecasts should be discarded when calculating covariance.
`horizons`	horizons for which forecast covariance should be calculated.
`zero`	if TRUE the covariance is calculated for a forecast of zero.
`trend`	if TRUE the covariance is calculated for a forecast of trend.
`estimation.sample`	portion of the sample to use for calculating the trend.
`compiled`	a logical indicating if compiled code should be used. (Usually true except for debugging.)
`...`	arguments passed to other methods.

Details

Calculate the forecast cov of obj relative to data. If obj is TSdata then the output data is used as the forecast. For other classes of obj TSmodel(obj) is used with data to produce a forecast. TSmodel() is also applied to each element of ... to extract a model. All models should work with data. If obj is a TSestModel and data is NULL then TSdata(obj) is used as the data. This is multiple applications of forecastCovSingleModel discard.before is an integer indicating the number of points in the beginning of forecasts to discard before calculating covariances. If it is the default, NULL, then the default (minimumStartupLag) will be used for each model and the default (1) will be used for trend and zero. If zero is TRUE then forecastCov is also calculated for a forecast of zero. If trend is TRUE then forecastCov is also calculated for a forecast of a linear trend using data to estimation.sample.

Value

A list with the forecast covariance for supplied models on the given sample. This is in the element `forecastCov` of the result. Other elements contain information in the arguments.

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model1 <- estVARXar(eg1.DSE.data.diff)
model2 <- estVARXls(eg1.DSE.data.diff)
z <-  forecastCov(model1, model2, data=trimNA(eg1.DSE.data.diff))
is.forecastCov(z)
```

`forecastCovReductionsWRTtrue`

Forecast covariance for different models

Description

Calculate the forecast covariance for different models.

Usage

```
forecastCovReductionsWRTtrue(true.model, rng=NULL,
                     simulation.args=NULL,
                     est.replications=2, pred.replications=2,
                     discard.before=10, horizons=1:12,quiet=FALSE,
                     estimation.methods=NULL,
                     criteria=NULL, compiled=.DSEflags()$COMPILED)
```

Arguments

`true.model`	An object of class TSmodel or TSestModel.
`discard.before`	An integer indicating the number of points in the beginning of forecasts to discard for calculating covariances.
`est.replications`	an interger indicating the number of times simulation and estimation are repeated.
`pred.replications`	an argument passed to `forecastCovWRTtrue`.
`simulation.args`	A list of any arguments which should be passed to simulate in order to simulate the true model.
`horizons`	Horizons for which forecast covariance should be calculated.
`rng`	If specified then it is used to set RNG.
`quiet`	If TRUE then some messages are not printed.
`estimation.methods`	a list as used by estimateModels.
`criteria`	a ...
`compiled`	a logical indicating if compiled code should be used. (Usually true except for debugging.)

Details

Calculate the forecasts cov of reduced models estimated from simulations of true.model with an estimation method indicated by estimation.methods. (estimation.methods is as in estimation.models BUT ONLY THE FIRST IS USED.) discard.before is an integer indicating 1+the number of points in the beginning of forecasts to discard for calculating forecast covariances. criteria can be a vector of criteria as in informationTests, (eg c("taic", "tbic") in which case the "best" model for each criteria is accounted separately. (ie. it is added to the beginning of the list of estimated models)

Value

A list ...

`forecastCovWRTtrue` *Compare Forecasts to True Model Output*

Description

Generate forecasts and compare them against the output of a true model.

Usage

```
forecastCovWRTtrue(models, true.model,
    pred.replications=1, simulation.args=NULL, quiet=FALSE, rng=NULL,
    compiled=.DSEflags()$COMPILED,
    horizons=1:12, discard.before=10, trend=NULL, zero=NULL,
    Spawn=if (exists(".SPAWN")) .SPAWN else FALSE)
is.forecastCovWRTdata(obj)
```

Arguments

`models`	A list of objects of class TSmodel.
`true.model`	An object of class TSmodel or TSestModel.
`discard.before`	An integer indicating the number of points in the beginning of forecasts to discard for calculating covariances.
`zero`	If TRUE then forecastCov is also calculated for a forecast of zero.
`trend`	If TRUE then forecastCov is also calculated for a forecast of a linear trend.
`pred.replications`	integer indicating the number of times simulated data is generated.
`simulation.args`	A list of any arguments which should be passed to simulate in order to simulate the true model.
`horizons`	Horizons for which forecast covariance should be calculated.
`rng`	If specified then it is used to set RNG.
`Spawn`	If TRUE then Splus For loops are used.
`quiet`	If TRUE then some messages are not printed.
`compiled`	a logical indicating if compiled code should be used. (Usually true except for debugging.)
`obj`	an object.

Details

The true model is used to generate data and for each generated data set the forecasts of the models are evaluated against the simulated data. If trend is not null it is treated as a model output (forecast) and should be the same dimension as a simulation of the models with simulation.args. If zero is not null a zero forecast is also evaluated. If simulating the true model requires input data then a convenient way to do this is for true.model to be a TSestModel. Otherwise, input data should be passed in simulation.args

Value

A list with the forecast covariance for supplied models on samples generated by the given true model. This is in the element `forecastCov` of the result. Other elements contain information in the arguments.

See Also

`forecastCovEstimatorsWRTdata` `simulate` `EstEval` `distribution` `MonteCarloSimulations`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
true.model <- estVARXls(eg1.DSE.data.diff) # A starting model TSestModel
data <- simulate(true.model)
models <- list(TSmodel(estVARXar(data)),TSmodel(estVARXls(data)))
z <-  forecastCovWRTtrue( models, true.model)
```

`forecast` *Forecast Multiple Steps Ahead*

Description

Calculate forecasts multiple steps ahead.

Usage

```
is.forecast(obj)
forecast(obj, ...)
## S3 method for class 'TSmodel':
forecast(obj, data,  horizon=36,
   conditioning.inputs=NULL,
   conditioning.inputs.forecasts=NULL, percent=NULL, ...)
## S3 method for class 'TSestModel':
forecast(obj, ...)
## S3 method for class 'TSdata':
forecast(obj, model, ...)
```

Arguments

`obj`	An object of a class for which a specific method is available.
`model`	An object of class TSmodel.
`data`	An object of class TSdata.
`conditioning.inputs`	A time series matrix or list of time series matrices to use as input variables.
`conditioning.inputs.forecasts`	A time series matrix or list of time series matrices to append to input variables for the forecast periods.
`horizon`	The number of periods to forecast.
`percent`	A vector indication percentages of the last input to use for forecast periods.
`...`	arguments passed to l().

Details

Calculate (multiple) forecasts from the end of data to a horizon determined either from supplied input data or the argument horizon (more details below). In the case of a model with no inputs the horizon is determined by the argument horizon. In the case of models with inputs, on which the forecasts are conditioned, the argument horizon is ignored (except when percent is specified) and the actual horizon is determined by the inputs in the following way: If inputs are not specified by optional arguments (as below) then the default will be to use inputData(data). This will be the same as the function l() unless inputData(data) is longer than outputData(data) (after NAs are trimmed from each separately). Otherwise, if conditioning.inputs is specified it is used for inputData(data). It must be a time series matrix or a list of time series matrices each of which is used in turn as inputData(data). The default above is the same as forecast(model, trimNA(data), conditioning.inputs=trimNA(inputData(data))) Otherwise, if conditioning.inputs.forecasts is specified it is appended to inputData(data). It must be a time series matrix or a list of time series matrices each of which is appended to inputData(data) and the concatenation used as conditioning.inputs. Both conditioning.inputs and conditioning.inputs.forecasts should not be specified. Otherwise, if percent is specified then conditioning.inputs.forecasts are set to percent/100 times the value of input corresponding to the last period of outputData(data) and used for horizon periods. percent can be a vector, in which case each value is applied in turn. ie c(90,100,110) would would give results for conditioning.input.forecasts 10 percent above and below the last value of input.

Value

The result is an object of class forecast which is a list with elements `model`, `horizon`, `conditioning.inputs`, `percent`, `pred` and `forecast`. The element `forecast` is a list with TSdata objects as elements, one for each element in the list conditioning.inputs. The element `pred` contains the one-step ahead forecasts for the preiods when output data is available. There is a plot method for this class.

See Also

`featherForecasts`, `horizonForecasts`

Examples

```
data("egJofF.1dec93.data", package="dse1")
model <- estVARXls(window(egJofF.1dec93.data, end=c(1985,12)))
pr <- forecast(model, conditioning.inputs=inputData(egJofF.1dec93.data))
#tfplot(pr) Rbug 0.90.1
is.forecast(pr)
```

`forecasts` *Extract Forecasts*

Description

Extract forecasts from and object.

Usage

```
forecasts(obj)
## S3 method for class 'forecast':
forecasts(obj)
## S3 method for class 'featherForecasts':
```

```
forecasts(obj)
## S3 method for class 'horizonForecasts':
forecasts(obj)
```

Arguments

obj	An object which contains forecasts.

Details

This generic method extracts the forecasts (only) from objects returned by other methods that calculate forecasts. Usually the objects returned by the methods which calculate forecasts contain additional information which is not returned by this extractor.

Value

The forecasts from an object which contains forecasts.

See Also

forecast

Examples

```
data("egJofF.1dec93.data", package="dse1")
model <- estVARXls(window(egJofF.1dec93.data, end=c(1985,12)))
pr <- forecast(model, conditioning.inputs=inputData(egJofF.1dec93.data))
z <- forecasts(pr)
```

---

generateSSmodel *Randomly generate a state space model*

Description

Randomly generate a state space model.

Usage

```
generateSSmodel(m,n,p, stable=FALSE)
```

Arguments

n,m,p	Input, state and output dimensions.
stable	TRUE or FALSE indicating if the model must be stable.

Details

Randomly generate a state space model. If stable is true then the largest root will have magnitude less than 1.0.

Value

An SS TSmodel.

Examples

```
z <-  generateSSmodel(2,3,1)
```

---

genMineData *Generate Data*

Description

Generate data for Monte Carlo experiments

Usage

```
genMineData(umodel, ymodel, uinput=NULL, sampleT=100,
  unoise=NULL, usd=1,ynoise=NULL, ysd=1, rng=NULL)
build.input.models(all.data, max.lag=NULL)
build.diagonal.model(multi.models)
```

Arguments

umodel	Model for input data.
ymodel	Model for output data.
sampleT	Number of periods of data to generate.
unoise	Input noise.
usd	Standard deviationof input noise.
ynoise	Output noise.
ysd	Standard deviation of output noise.
rng	RNG setting.
multi.models	
all.data	
max.lag	
uinput	

Details

This function generates test data using specified models. umodel is used to generate data input data and ymodel is used to generate data corresponding output data. The result of umodel is used as input to ymodel so the input dimension of ymodel should be the output dimension of umodel. Typically the ymodel would be degenerate in some of the input variables so the effective inputs are a subset. If noise is NULL then an normal noise will be generated by simulate. This will be iid N(0,I). The RNG will be set first if it is specified. If unoise or ynoise are specified they should be as expected by simulate for the specified umodel and ymodel.

`genMineData` uses `build.input.models` which makes a list of univariate models, one for each series in `inputData(data)` and `build.diagonal.model` which builds one diagonal model from a list of models returned by `build.input.models`. It uses the AR part only.

Value

A TSdata object.

See Also

simulate

Examples

```
data("eg1.DSE.data.diff", package="dse1")
umodel <- build.diagonal.model(
        build.input.models(eg1.DSE.data.diff, max.lag=2))
z  <- TSdata(output=outputData(eg1.DSE.data.diff),
             input = inputData(eg1.DSE.data.diff))
ymodel <- TSmodel(estVARXls(z, max.lag=3))
sim.data <- genMineData(umodel, ymodel)
```

---

horizonForecastsCompiled

Calculate forecasts at specified horizons

Description

Calculate forecasts at specified horizons.

Usage

```
horizonForecastsCompiled(obj, data, horizons=1:4,
       discard.before=minimumStartupLag(obj))
## S3 method for class 'SS':
horizonForecastsCompiled(obj, data, horizons=1:4,
      discard.before=minimumStartupLag(obj))
## S3 method for class 'ARMA':
horizonForecastsCompiled(obj, data, horizons=1:4,
       discard.before=minimumStartupLag(obj))
```

Arguments

obj	see horizonForecasts.
data	see horizonForecasts.
horizons	see horizonForecasts.
discard.before	see horizonForecasts.
...	see horizonForecasts.

Details

Internal function not to be called by users. See horizonForecasts.

Value

See horizonForecasts.

See Also

horizonForecasts

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(eg1.DSE.data.diff)
z <-  horizonForecasts(model, eg1.DSE.data.diff)
```

---

horizonForecasts *Calculate forecasts at specified horizons*

Description

Calculate forecasts at specified horizons.

Usage

```
is.horizonForecasts(obj)
horizonForecasts(obj, ...)
## S3 method for class 'TSmodel':
horizonForecasts(obj, data, horizons=1:4,
     discard.before=minimumStartupLag(obj), compiled=.DSEflags()$COMPILED, .
## S3 method for class 'TSestModel':
horizonForecasts(obj, data=NULL, ...)
## S3 method for class 'TSdata':
horizonForecasts(obj, model, ...)
## S3 method for class 'forecastCov':
horizonForecasts(obj,horizons=NULL,
     discard.before=NULL, ...)
```

Arguments

`obj`	an object of class TSmodel, TSdata, or TSestModel.
`model`	an object of class TSmodel.
`data`	an object of class TSdata
`horizons`	a vector of integers indicating the horizon at which forecasts should be produced.
`discard.before`	period before which forecasts are not calculated.
`compiled`	if TRUE compiled code is called.
`...`	arguments passed to other methods.

Details

Calculate multiple 'horizon'-step ahead forecasts ie. calculate forecasts but return only those indicated by horizons. Thus, for example, the result of horizonForecasts(model, data horizons=c(1,5)) would be the one-step ahead and five step ahead forecasts.

Value

The result is a list of class horizonForecasts with elements model (a TSmodel), data, horizons, discard.before, and horizonForecasts. horizonForecasts is an array with three dimension: `c(length(horizons), dim(model$data))`. Projections are not calculated before discard.before or after the end of outputData(data). Each horizon is aligned so that horizonForecasts[h,t,] contains the forecast for the data point outputData(data)[t,] (from horizon[h] periods prior).

See Also

featherForecasts

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(eg1.DSE.data.diff)
z <-  horizonForecasts(model, eg1.DSE.data.diff)
```

---

is.forecastCovEstimatorsWRTdata.subsets

Check Inheritance

Description

Check inheritance.

Usage

```
is.forecastCovEstimatorsWRTdata.subsets(obj)
```

Arguments

obj	Any object.

Details

This tests if an object isnherits from `forecastCovEstimatorsWRTdata.subsets`. This type of object code be generated in different ways but the only current example is `stripMine`.

Value

logical

See Also

stripMine

`mineStepwise` *Mine Stepwise*

Description

mineStepwise

Usage

```
mineStepwise(data, essential.data=1,
  method="efroymson", f.crit=2, intercept=TRUE,
  subtract.means=FALSE,  standardize=FALSE,
  lags.in=6, lags.out=6, trend=FALSE, plot.=TRUE)
```

Arguments

`data`	TSdata
`essential.data`	An integer vector indication important data.
`method`	method to pass to stepwise.
`f.crit`	See details.
`intercept`	See details.
`subtract.means`	See details.
`standardize`	See details.
`lags.in`	See details.
`lags.out`	See details.
`trend`	See details.
`plot.`	See details.

Details

This documentation is out of date. Data should be of class TSdata. essential.data must have length 1. standardize and subtract means ... The result is a list with the results of stepwise,..., and several vectors indicating information about the columns of the matrix passed to stepwise: io.indicator indicating an input (FALSE) or output (TRUE) variable v.indicator indicating which series lag.indicator indicating the lag s.input.indicator and s.output.indicator are logic matrices `length(stepwise$rss)` by m and p respectively indicating if a series is included for each element of rss.

Value

x

`minForecastCov` *Minimum Forecast Cov Models*

Description

Extract the minimum forecastCov at each horizon

Usage

```
minForecastCov(obj, series=1, verbose=TRUE)
```

Arguments

`obj`	An object as returned by stripMine.
`series`	An indicator of the series which are to be used as the bases for selection.
`verbose`	If true additional information is printed.

Details

Select the min covariance (for series only!) at each horizon and print. The returned object is a vector indicating the element of forecastCov which was the min at each horizon. It is suitable as an argument to plot eg: tfplot(obj, select.cov=minForecastCov(obj)) The results of this plot are similar to the default results of tfplot(selectForecastCov(obj)). Covariance information and information about the horizon where the model is optimal are given.

Value

The returned object is a vector indicating the element of forecastCov which was the min at each horizon.

See Also

`selectForecastCov`, `excludeForecastCov`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
z <- stripMine(eg1.DSE.data.diff, essential.data=c(1,2),
                   estimation.methods=list(estVARXls=list(max.lag=3)))
z <-  minForecastCov(z)
```

`minimumStartupLag` *Starting Periods Required*

Description

Number of Starting Periods Required for a Model

Usage

```
minimumStartupLag(model)
## S3 method for class 'SS':
minimumStartupLag(model)
## S3 method for class 'ARMA':
minimumStartupLag(model)
## S3 method for class 'TSestModel':
minimumStartupLag(model)
startShift(model,data, y0=NULL)
```

Arguments

`model`	A TSmodel or object containing a TSmodel.
`data`	A TSdata object.
`y0`	initial condition ...

Details

For many time series models several starting data points are required before output from the model can be calculated (or makes sense). This generic function extracts or calculates that number of periods.

Value

An integer.

Note

There is some redundancy between this and startShift which should be cleaned up.

See Also

`TSmodel`

`MonteCarloSimulations` *Generate simulations*

Description

Run multiple simulations

Usage

```
is.MonteCarloSimulations(obj)
MonteCarloSimulations(model, simulation.args=NULL,
       replications=100, rng=NULL, quiet =FALSE, ...)
## Default S3 method:
MonteCarloSimulations(model, simulation.args = NULL,
            replications = 100, rng = NULL, quiet =FALSE, ...)
## S3 method for class 'TSmodel':
MonteCarloSimulations(model, simulation.args=NULL,
      replications=100, rng=NULL, quiet=FALSE, ...)
## S3 method for class 'TSestModel':
MonteCarloSimulations(model, simulation.args=NULL,
       replications=100, rng=NULL, quiet=FALSE, ...)
## S3 method for class 'EstEval':
MonteCarloSimulations(model, simulation.args=NULL,
        replications=100, rng=getRNG(model),  quiet=FALSE, ...)
## S3 method for class 'MonteCarloSimulations':
MonteCarloSimulations(model,
   simulation.args=NULL, replications=100, rng=getRNG(model),  quiet=FALSE,
```

Arguments

`model`	an object from which a model can be extracted. The model must have an associated `simulation` method (e.g. a TSmodel).
`simulation.args,`	A list of arguments in addition to model which are passed to simulate.
`replications`	The number of simulations.
`rng`	The RNG and starting seed.
`quiet`	logical indicating if printing and many warning messages should be suppressed.
`obj`	an object.
`...`	arguments passed to other methods.

Details

This function runs many simulations using `simulate`. Often it not be necessary to do this since the seed can be used to reproduce the sample and many functions for testing estimation methods, etc., will produce samples as they proceed. This function is useful for verification and for looking at the stochastic properties of the output of a model. If `model` is an object of class `EstEval` or `simulation` then the model and the seed!!! are extracted so the same sample will be generated. The default method expects the result of `simulate(model)` to be a matrix. There is a `tfplot` method (time series plots of the simulations) and a `distribution` method for the result. The latter plots kernel estimates of the distribution of the simulations at specified periods.

Value

A list of simulations.

See Also

simulate EstEval distribution forecastCovWRTtrue

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(eg1.DSE.data.diff)
z <-  MonteCarloSimulations(model, simulation.args=list(sampleT=100))
tfplot(z)
distribution(z)
```

---

nseriesfeatherForecasts

Number of Series

Description

Return the number of series.

Usage

```
## S3 method for class 'featherForecasts':
nseries(x)
## S3 method for class 'MonteCarloSimulations':
nseriesInput(x)
## S3 method for class 'MonteCarloSimulations':
nseriesOutput(x)
```

Arguments

x	A featherForecasts object.

Details

See the generic method.

Value

An integer.

outOfSample.forecastCovEstimatorsWRTdata

Calculate Out-of-Sample Forecasts

Description

Calculate out-of-sample forecasts.

Usage

```
outOfSample.forecastCovEstimatorsWRTdata(data, zero=FALSE, trend=FALSE,
                    estimation.sample=.5, horizons=1:12,quiet=FALSE,
                    estimation.methods=NULL, compiled=.DSEflags()$COMPILED)
```

Arguments

`data`	an object of class TSdata.
`estimation.methods`	a list as used by estimateModels.
`zero`	if TRUE then forecastCov is also calculated for a forecast of zero.
`trend`	if TRUE then forecastCov is also calculated for a forecast of a linear trend.
`estimation.sample`	indicates the portion of the data to use for estimation. If estimation.sample is an integer then it is used to indicate the number of points in the sample to use for estimation. If it is a fracton it is used to indicate the portion of points to use for estimation. The remainder of the sample is used for evaluating forecasts.
`horizons`	horizons for which forecast covariance should be calculated.
`quiet`	if TRUE then estimation information is not printed.
`compiled`	a logical indicating if compiled code should be used. (Usually true except for debugging.)

Details

The data is slpit into a sub-sample used for estimation and another sub-sample used for calculating the forecast covariance.

Value

An object as returned by forecastCovEstimatorsWRTdata.

See Also

`forecastCovEstimatorsWRTdata`, `forecastCovEstimatorsWRTtrue`, `estimateModels`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
z <-  outOfSample.forecastCovEstimatorsWRTdata(eg1.DSE.data.diff,
          estimation.methods=list(estVARXls=list(max.lag=4)))
```

periodsMonteCarloSimulations *Tframe or Number of Periods*

Description

Return the number of periods or the tframe.

Usage

```
## S3 method for class 'MonteCarloSimulations':
periods(x)
## S3 method for class 'MonteCarloSimulations':
tframe(x)
```

Arguments

x	A MonteCarloSimulations object.

Details

See the generic method.

Value

An integer or a tframe object.

permute *Permute*

Description

Return matrix with rows indicating all possible selections of elements from seq(M). 0 in the result indicates omit. M is usually a positive integer. M=0 gives NULL. Neg. M give -permute(abs(M)).

Usage

```
permute(M)
```

Arguments

M	An integer.

Value

A matrix.

Examples

```
permute(4)
```

phasePlots *Calculate Phase Plots*

Description

Calculate phase plots

Usage

```
phasePlots(data, max.lag=1,diff=FALSE)
```

Arguments

data	A matrix, time series matrix, or an object of class TSdata.
max.lag	The maximum number of shifts to plot
diff	If TRUE the data is plotted against the difference with lagged values.

Details

Non-linearities may show up as a non-linear surface, but this is a projection so, for example, a spherical space would not show up. Some sort of cross-section window would show this but require even more plots. A good statistical test would be better!

Value

None

Side Effects

A plot of (the phase space) the data against (differenced) lagged values is produced.

Examples

```
data("egJofF.1dec93.data", package="dse1")
phasePlots(egJofF.1dec93.data)
```

plot.mineStepwise *Plot Mine Stepwise Object*

Description

plot.mineStepwise

Usage

```
## S3 method for class 'mineStepwise':
plot(x, ...)
```

Arguments

x	Object returned by mineStepwise.
...	(further arguments, currently disregarded).

Value

None

Side Effects

A plot

See Also

mineStepwise

---

print.estimatedModels

Print Specific Methods

Description

See the generic function description.

Usage

```
## S3 method for class 'estimatedModels':
print(x, digits=options()$digits, ...)
## S3 method for class 'EstEval':
print(x, digits=options()$digits, ...)
## S3 method for class 'forecastCov':
print(x, digits=options()$digits, ...)
## S3 method for class 'forecastCovEstimatorsWRTdata.subsets':
print(x, digits=options()$digits, ...)
## S3 method for class 'forecastCovEstimatorsWRTtrue':
print(x, digits=options()$digits, ...)
## S3 method for class 'MonteCarloSimulations':
print(x, digits=options()$digits, ...)
```

Arguments

x	an object to be printed.
digits	a non-null value is used to indicate the number of significant digits. If digits is NULL then the value of digits specified by options is used.
...	(further arguments, currently disregarded).

See Also

print summary

`roots.estimatedModels` *Roots Specific Methods*

Description

See the generic function description.

Usage

```
## S3 method for class 'estimatedModels':
roots(obj, digits=options()$digits, mod =FALSE, ...)
## S3 method for class 'forecastCovEstimatorsWRTtrue':
roots(obj, digits=options()$digits,
   mod=FALSE, ...)
## S3 method for class 'coefEstEval':
roots(obj, criterion.args=NULL, ...)
## S3 method for class 'rootsEstEval':
roots(obj, ...)
## S3 method for class 'TSestModelEstEval':
roots(obj, criterion.args=NULL, ...)
## S3 method for class 'TSmodelEstEval':
roots(obj, criterion.args=list(randomize = TRUE), ...)
```

Arguments

`obj`	an object from which roots are to be extracted or calculated and printed.
`digits`	an integer indicating the number of significant digits to be printed (passed to the print method).
`mod`	if TRUE the modulus of the roots is calculated. Otherwise, a complex value may result.
`criterion.args`	arguments to be passed to this method when it is called by `EstEval`.
`...`	arguments to be passed to other methods.

Details

The methods ***.ee are intended mainly to be called from EstEval as criterion for evaluating an estimation method.

See Also

`roots` `stability` `EstEval`

`selectForecastCov` *Select Forecast Covariances Meeting Criteria*

Description

Select forecast covariances meeting given criteria.

Usage

```
selectForecastCov(obj, series=1,
select.cov.best=1,
select.cov.bound=NULL,
ranked.on.cov.bound=NULL,
verbose=TRUE)
```

Arguments

`obj`	an object as returned by stripMine.
`series`	an indication of series to which the tests should be applied.
`select.cov.best`	the number of 'best' forecasts to select.
`select.cov.bound`	a bound to use as criteria for selection.
`ranked.on.cov.bound`	see details.
`verbose`	if verbose=TRUE then summary results are printed.

Details

Select models with forecast covariance for `series` meeting criteria. The default `select.cov.best=1` selects the best model at each horizon. `select.cov.best=3` would select the best 3 models at each horizon. If `select.cov.bound` is not `NULL` then `select.cov.best` is ignored and any model which is better than the bound at all horizons is selected. `select.cov.bound` can be a vector of the same length as `series`, in which case corresponding elements are applied to the different series. Any model which is better than the bound at all horizons is selected. `ranked.on.cov.bound` is is used if it is not NULL and `select.cov.bound` is `NULL`. In this case `select.cov.best` is ignored. `ranked.on.cov.bound` should be a positive integer. The forecast covariances are ranked by there maximum over the horizon and the lowest number up to `ranked.on.cov.bound` are selected. This amounts to adjusting the covariance bound to allow for the given number of models to be selected. If `series` is a vector the results are the best up to the given number on any series! `select.cov.bound` can be a vector of the same length as `series`, in which case corresponding elements are applied to the different series. If `verbose=TRUE` then summary results are printed. The returned result is a `forecastCov` object like obj, but filtered to remove models which do not meet criteria.

Value

The returned result is a forecastCov object like obj, but filtered to remove models which do not meet criteria.

See Also

minForecastCov, excludeForecastCov

Examples

```
data("eg1.DSE.data.diff", package="dse1")
z <- stripMine(eg1.DSE.data.diff, essential.data=c(1,2),
                  estimation.methods=list(estVARXls=list(max.lag=3)))
z <-  selectForecastCov(z)
tfplot(selectForecastCov(z, select.cov.bound=20000))
tfplot(selectForecastCov(z, select.cov.best=1))
```

---

seriesNamesInput.forecast

TS Input and Output Specific Methods

Description

See the generic function description.

Usage

```
## S3 method for class 'forecast':
seriesNamesInput(x)
## S3 method for class 'featherForecasts':
seriesNamesInput(x)
## S3 method for class 'MonteCarloSimulations':
seriesNamesInput(x)

## S3 method for class 'forecast':
seriesNamesOutput(x)
## S3 method for class 'featherForecasts':
seriesNamesOutput(x)
## S3 method for class 'MonteCarloSimulations':
seriesNamesOutput(x)
```

Arguments

x	an object from which to extract the names of the input or output series.

---

shockDecomposition *Shock Decomposition*

Description

Graphs of the effect of shocks are plotted.

Usage

```
shockDecomposition(model, horizon=30, shock=rep(1,horizon))
```

Arguments

model	An object of class TSmodel or TSestModel.
horizon	The number of periods for which to calculate the effect of shocks.
shock	data to be used model output. See details.

Details

All output data is set to zero and then each output in turn is switched to a value of shock (default 1.0) for all periods.

Value

None

Side Effects

Graphs of the effect of shocks are plotted.

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(eg1.DSE.data.diff)
shockDecomposition(model)
```

---

stripMine *Select a Data Subset and Model*

Description

Select a data subset and model.

Usage

```
stripMine(all.data, essential.data=1,
                    estimation.sample=.5,
                    discard.before=1, horizons=1:12,quiet=FALSE,
                    estimation.methods=NULL,
                    step.size=NULL)
```

Arguments

all.data	An object of class TSdata.
essential.data	A vector indicating the important series.
estimation.sample	The portion of the data to use for estimation.
discard.before	Period before which data should be disacrded when calculating the forecast covariances.
horizons	Forecast horizons which should be considered.

`quiet` If T then estimation information is not printed. quiet=TRUE may also have to be set in the arguments to estimation methods.

`estimation.methods`
A list indicating the model estimation method to use. The list should contain one element. The name of the element indicates the estimation method to use and the value of the element is a list of arguments to pass to the estimation method.

`step.size` An integer indicting how many dta subset/model steps should be attempted. This may be necessary to accommodate memory constraints on the system. (see below.)

Details

Calculate the predictions cov for essential.data of models estimated with estimation methods indicated by estimation.methods. estimation.methods is a list with syntax similar to programs for comparing estimation methods (eg. estimateModels), BUT ONLY THE FIRST element (estimation method) is considered. Essential.data indicates the subset of output variables to included in all models. It should be a vector of the indices. All possible combinations of input series and other output series data are considered. If omitted, essential.data is taken as the first output series. Only forecast covariances for essential data are returned. discard.before is an integer indicating 1+the number of points in the beginning of predictions to discard for calculating prediction covariances. estimation.sample indicates the portion of the data to use for estimation. If estimation.sample is an integer then it is used to indicate the number of points in the sample to use for estimation. If it is a fracton it is used to indicate the portion of points to use for estimation. The remainder of the sample is used for evaluating predictions. If step.size is NULL then all possible data permutations are attempted. Because S has a hard-coded limit in the number of synchronize calls this is not always possible (For loops call synchronize.) An error message: Error in synchronize(1): No room in database table If step.size is not NULL it should be a positive integer. In this case variable permutions are divided up into steps of the given size. The result returned by the function can be used to continue from the last step: intermediate.result <- stripMine(data, ...) intermediate.result <- stripMine(intermediate.result) intermediate.result <- stripMine(intermediate.result) result <- stripMine(intermediate.result) This can be done either interactively or in a batch process, but cannot be done in a function because the database table is not cleared until the top level expression is complete. The class of an intermediate result is stripMine.intermediate.result and the class of the final result is c('forecastCovEstimatorsWRTdata.subsets', 'forecastCov') If the final result is used in a call to stripMine then it is just returned, so extra calls do not cause errors and are very quick. This is useful when you are too lazy to calculate the exact number of steps.

Value

The returned result contains a list (`forecastCov`) of the forecast covariance on the essential data for the various models and data subsets. It can be plotted with the generic function tfplot. Additional information in the result comes from the function arguments.

See Also

`estBlackBox4`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
z <- stripMine(eg1.DSE.data.diff,
    estimation.methods=list(bft=list(max.lag=2, verbose=FALSE)))
```

`summary.estimatedModels` *Summary Specific Methods*

Description

See the generic function description.

Usage

```
## S3 method for class 'estimatedModels':
summary(object, ...)
## S3 method for class 'TSestModelEstEval':
summary(object, ...)
## S3 method for class 'TSmodelEstEval':
summary(object, ...)
## S3 method for class 'EstEval':
summary(object, ...)
## S3 method for class 'forecastCov':
summary(object, horizons=object$horizons,
    series=seq(nseriesOutput(object$data)), ...)
## S3 method for class 'forecastCovEstimatorsWRTdata.subsets':
summary(object, ...)
## S3 method for class 'forecastCovEstimatorsWRTtrue':
summary(object,
    digits=options()$digits, ...)
## S3 method for class 'MonteCarloSimulations':
summary(object, series=NULL, periods=1:3, ...)
## S3 method for class 'coefEstEval':
summary(object, verbose=TRUE, ...)
## S3 method for class 'rootsEstEval':
summary(object, verbose=TRUE, ...)

## S3 method for class 'summary.estimatedModels':
print(x, digits=options()$digits, ...)
## S3 method for class 'summary.TSestModelEstEval':
print(x, digits=options()$digits, ...)
## S3 method for class 'summary.TSmodelEstEval':
print(x, digits=options()$digits, ...)
## S3 method for class 'summary.EstEval':
print(x, digits=options()$digits, ...)
## S3 method for class 'summary.forecastCov':
print(x, digits=options()$digits, ...)
## S3 method for class 'summary.forecastCovEstimatorsWRTdata.subsets':
print(x,
   digits=options()$digits, ...)
## S3 method for class 'summary.forecastCovEstimatorsWRTtrue':
print(x,
   digits=options()$digits, ...)
## S3 method for class 'summary.MonteCarloSimulations':
print(x, digits=options()$digits, ...)
```

```
## S3 method for class 'summary.coefEstEval':
print(x, digits=options()$digits, ...)
## S3 method for class 'summary.rootsEstEval':
print(x, digits=options()$digits, ...)
```

Arguments

object	an object for which a summary is to be printed.
x	an object for which a summary is to be printed.
digits	a non-null value is used to indicate the number of significant digits. If digits is NULL then the value of digits specified by options is used.
horizons	optional integer vector indicating horizons at which the summary should be calculated.
series	The series which should be plotted. The default NULL gives all series.
periods	optional integer vector indicating periods at which the summary should be calculated.
verbose	logical indicating if a longer summary should be produced.
...	arguments passed to other methods.

See Also

summary print

---

testEqual.estimatedModels

Specific Methods for Testing Equality

Description

See the generic function description.

Usage

```
## S3 method for class 'estimatedModels':
testEqual(obj1, obj2, fuzz = 0)
## S3 method for class 'EstEval':
testEqual(obj1, obj2, fuzz=0)
## S3 method for class 'forecast':
testEqual(obj1, obj2, fuzz=1e-14)
## S3 method for class 'forecastCov':
testEqual(obj1, obj2, fuzz=1e-14)
## S3 method for class 'horizonForecasts':
testEqual(obj1, obj2, fuzz=1e-14)
## S3 method for class 'MonteCarloSimulations':
testEqual(obj1, obj2, fuzz=1e-16)
```

Arguments

obj1	an object which is to be compared with the second object.
obj2	an object which is to be compared with the first object.
fuzz	tolerance for numerical comparisons. Values within fuzz will be considered equal.

See Also

testEqual

---

tfplot.coefEstEval *Specific tfplot methods for coefEstEval (EstEval) objects*

Description

See the generic function description.

Usage

```
## S3 method for class 'coefEstEval':
tfplot(x, cumulate=TRUE, norm=FALSE, bounds=TRUE,
    invert=FALSE, Sort=FALSE, graphs.per.page = 5, ...)
```

Arguments

x	an object for which a tfplot is to be produced.
cumulate	logical indicating if the cumulative average of roots should be plotted
invert	logical indicating if the inverse of roots should be plotted
Sort	logical indicating if the roots should be sorted.
graphs.per.page	integer indicating number of graphs to place on a page.
norm	logical indicating if the euclidean norm of roots should be plotted (square root of the sum of squared roots).
bounds	logical indicating if estimated one standard error bounds should be plotted around the lines for the true roots.
...	arguments passed to other methods.

Details

If cumulate is true the cumulative average is plotted. If norm is true the norm is used, each parameter is plotted. If invert is true the reciprical is used (before cumulating). If Sort is true then sort is applied (before ave). This is not usually recommended but of interest with estimation methods like black.box which may not return parameters of the same length or in the same order. Plotting the true lines only makes sense if truth is the same length as result (and sometimes not even then).

See Also

tfplot EstEval

tfplot.forecastCov *Plots of Forecast Variance*

Description

Generate plots of forecast variance calculated by forecastCov.

Usage

```
## S3 method for class 'forecastCov':
tfplot(x, ...,
   series = 1:dim(x$forecastCov[[1]])[2],
   select.cov = 1:length(x$forecastCov), select.true =TRUE,
   select.zero =TRUE, select.trend =TRUE, y.limit = NULL, line.labels =FALSE
   lty = NULL, Legend = NULL, Title = NULL,
   graphs.per.page = 5, mar=par()$mar, reset.screen=TRUE)
## S3 method for class 'forecastCovEstimatorsWRTdata':
tfplot(x,
    series=1:dim(x$forecastCov[[1]])[2],
    select.cov=1:length(x$forecastCov),
    select.zero=TRUE, select.trend=TRUE,
    graphs.per.page = 5, mar=par()$mar, reset.screen=TRUE, lty=NULL, ...)
```

Arguments

x	The result of forecastCov.
series	integer or string indicating the series which should be plotted.
select.cov	logical indicating that for the case of multiple models select the covariance to be plotted.
select.true	logical indicating that results from the forecast of the true model (if available) should be plotted.
select.zero	logical indicating that results from a forecast of zero should be plotted.
select.trend	logical indicating that results from a forecast of trend should be plotted.
graphs.per.page	The number of graphs to put on a page.
mar	plot margins (see par).
reset.screen	logical indicating if the plot window should be cleared before starting.
lty	see details.
Legend	optional legend passed to legend.
Title	optional legend passed to title (but see details).
y.limit	optional limit on the y scale. Covariance values larger than y.limit will not be shown.
line.labels	logical indicating line labels should be printed.
...	For forecastCov objects this allows additional objects to be plotted. For forecastCovEstimatorsWRTdata ... are passed to other methods.

Details

This function produces plots of the variance at different horizons. Output graphics can be paused between pages by setting par(ask=TRUE). If lty is NULL (default) it is set to seq(length(select.cov) +select.true+select.zero+select.trend), and corrected if these are TRUE but not in the object.

The `Title` is not put on the plot if the global option PlotTitles is FALSE. This can be set with `options(PlotTitles=FALSE)`. This provides a convenient mechanism to omit all titles when the title may be added separately (e.g. in Latex).

Value

None

See Also

`plot`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(eg1.DSE.data.diff)
z <- forecastCov(model, data=eg1.DSE.data.diff)
tfplot(z)
```

`tfplot.MonteCarloSimulations`

Generate plots of Monte Carlo simulations

Description

Generate plots of Monte Carlo simulations.

Usage

```
## S3 method for class 'MonteCarloSimulations':
tfplot(x,
tf=tframe(x$simulations), start=tfstart(tf), end=tfend(tf),
series=seq((dim(x$simulations)[2])),
select.simulations=seq(dim(x$simulations)[3]),
graphs.per.page=5, mar=par()$mar, ...)
```

Arguments

`x`	The result of MonteCarloSimulations.
`tf`	The time frame for plots. see `tfplot`.
`start`	The starting period for plots, taken from tf by default.
`end`	The ending period for plots, taken from tf by default.
`series`	The series which should be plotted. The default NULL gives all series.
`select.simulations`	Vector of integers indicating the simulations which should be plotted. The default plots all simulations.

`graphs.per.page`
: The number of graphs to put on a page.

`mar`
: Plot margins (see `par`).

`...`
: arguments passed to other methods.

Details

This function produces plots of the simulated series. Output graphics can be paused between pages by setting par(ask=TRUE).

Value

None

See Also

`distribution.MonteCarloSimulations`

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(eg1.DSE.data.diff)
z <-  MonteCarloSimulations(model)
tfplot(z)
```

---

tfplot.rootsEstEval

Specific tfplot methods for rootsEstEval (EstEval) objects

Description

See the generic function description.

Usage

```
## S3 method for class 'rootsEstEval':
tfplot(x, ...)
## S3 method for class 'rootsEstEval':
plot(x, complex.plane=TRUE, cumulate=TRUE, norm=FALSE,
   bounds=TRUE, transform=NULL, invert=FALSE, Sort=TRUE, ...)
```

Arguments

`x`
: an object for which a tfplot is to be produced.

`complex.plane`
: logical indicating if the plot should be on the complex plane.

`cumulate`
: logical indicating if the cumulative average of roots should be plotted

`invert`
: logical indicating if the inverse of roots should be plotted

`Sort`
: logical indicating if the roots should be sorted.

`...`
: arguments passed to other methods.

norm	logical indicating if the euclidean norm of roots should be plotted (square root of the sum of squared roots).
bounds	logical indicating if estimated one standard error bounds should be plotted around the lines for the true roots.
transform	an optional string indicating the name of a function which should be applied to the roots before plotting.

Details

If complex.plane is TRUE then all results are plotted on a complex plane and the arguements cumulate and Sort do not apply. If complex.plane is FALSE then a sequential plot of the real and imaginary parts is produced. If cumulate is true the cumulative average is plotted. If mod is true the modulus is used, otherwise real and imaginary are separated. if invert is true the reciprical is used (before cumulating). if Sort is true then sort is applied (before cumulate but after mod) by the Re part of the root. Some grouping is usually necessary since roots are not in an obvious order but sorting by the real part of the roots could be improved upon.

See Also

tfplot EstEval

---

tfplot.TSdata.ee *Specific Methods for tfplot*

Description

See the generic function description.

Usage

```
## S3 method for class 'TSmodelEstEval':
tfplot(x, graph.args=NULL,
                    criterion ="coef", criterion.args=NULL, ...)
## S3 method for class 'TSestModelEstEval':
tfplot(x, graph.args=NULL,
                    criterion ="coef", criterion.args=NULL, ...)
## S3 method for class 'featherForecasts':
tfplot(x, tf=NULL, start=tfstart(tf), end=tfend(tf),
    series=seq(nseries(x)),
    Title="Predictions (dotted) and actual data (solid)",
    ylab=seriesNamesOutput(x),
    graphs.per.page=5, mar=par()$mar, reset.screen=TRUE, ...)
## S3 method for class 'forecast':
tfplot(x, tf=NULL, start=tfstart(tf), end=tfend(tf),
    series = seq(length=nseriesOutput(x$data)),
    Title="Predictions (dotted) and actual data (solid)",
    ylab = seriesNamesOutput(x$data),
    graphs.per.page=5, mar=par()$mar, reset.screen=TRUE, ...)
## S3 method for class 'EstEval':
tfplot(x, tf=NULL, start=tfstart(tf), end=tfend(tf),
    truth= if(is.TSdata(x$truth)) outputData(x$truth) else x$truth,
```

```
    series = seq(length=nseries(truth)),
    Title="Estimated (and true) results",
    ylab = seriesNames(truth), remove.mean = FALSE,
    graphs.per.page=5, mar=par()$mar, reset.screen=TRUE, ...)
## S3 method for class 'horizonForecasts':
tfplot(x, tf=NULL, start=tfstart(tf), end=tfend(tf),
     series=seq(length=nseriesOutput(x$data)),
     Title="Predictions (dotted) and actual data (solid)",
     ylab=seriesNamesOutput(x$data),
     graphs.per.page=5, mar=par()$mar, reset.screen=TRUE, ...)
## S3 method for class 'multiModelHorizonForecasts':
tfplot(x,
     tf=NULL, start=tfstart(tf), end=tfend(tf), series=NULL, ...)
```

Arguments

`x`	an object for which a tfplot is to be produced.
`tf`	see `tfplot`.
`start`	see `tfplot`.
`end`	see `tfplot`.
`truth`	true value which will be plotted along with estimates.
`Title`	string of characters to use for title.
`remove.mean`	logical indicating if means should be removed before plotting results.
`ylab`	vector of strings for y axis labelling.
`graphs.per.page`	integer indicating number of graphs to place on a page.
`reset.screen`	logical indicating if the plot window should be cleared before starting.
`series`	integer or string indicating the series which should be plotted.
`mar`	plot margins. See `par`.
`graph.args`	list of graphics arguments eventually passed to `plot`. See `par`.
`criterion`	criterion which should be used to extract something from the object which will then be plotted. See `EstEval`.
`criterion.args`	arguments to be passed to `criterion`.
`...`	arguments passed to other methods.

See Also

`tfplot` `EstEval`

totalForecastCov *Sum covariance of forecasts across all series*

Description

Sum covariance of forecasts across all series.

Usage

```
totalForecastCov(obj, select=NULL)
```

Arguments

obj	An object as returned by forecastCov.
select	Series to be select for summation. With the default all series are selected.

Value

An object similar to that returned by forecastCov, with the covariance summed over all selected series.

Examples

```
data("eg1.DSE.data.diff", package="dse1")
model1 <- estVARXar(eg1.DSE.data.diff)
model2 <- estVARXls(eg1.DSE.data.diff)
z <-  totalForecastCov(forecastCov(model1, model2,
                         data=trimNA(eg1.DSE.data.diff)))
```

TSdata.forecastCov *TS Extractor Specific Methods*

Description

See the generic function description.

Usage

```
## S3 method for class 'forecastCov':
TSdata(data, ...)
## S3 method for class 'coefEstEval':
TSestModel(obj)
## S3 method for class 'forecastCov':
TSmodel(obj, select=1, ...)
## S3 method for class 'coefEstEval':
TSmodel(obj, ...)
```

Arguments

`data`	an object from which to extract the TSdata.
`obj`	an object from which to extract the TSmodel or TSestModel.
`select`	an integer indicating which of multiple models to extract.
`...`	arguments to be passed to other methods.

See Also

`TSdata` `TSestModel` `TSmodel`

Index