

Package ‘EvalEst’

October 18, 2009

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), dse (>= 2007.10-1)

Version 2009.10-2

Date 2009-10-16

LazyLoad yes

License GPL-2 | file LICENSE

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

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

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

Repository CRAN

Date/Publication 2009-10-18 10:03:32

R topics documented:

coef.TSmodelEstEval	2
distribution	3
distribution.coefEstEval	4
EstEval	5
generateSSmodel	7
genMineData	7
MonteCarloSimulations	9
nseries.MonteCarloSimulations	10
periodsMonteCarloSimulations	11
print.estimatedModels	11
roots.coefEstEval	12
seriesNamesInput.MonteCarloSimulations	13
summary.EstEval	13

testEqual.EstEval	14
tfplot.coefEstEval	15
tfplot.MonteCarloSimulations	16
tfplot.rootsEstEval	17
tfplot.TSdata.ee	18
TSdata.coefEstEval	19

Index	20
--------------	-----------

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](#)

distribution *Generate distribution plots of Monte Carlo simulations*

Description

Generate distribution plots of Monte Carlo simulations.

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

Arguments

<code>obj</code>	The result of MonteCarloSimulations.
<code>bandwidth</code>	passed to <code>density</code> or <code>ksmooth</code> .
<code>series</code>	The series which should be plotted. The default gives all series.
<code>select.inputs</code>	series to be plotted. (passed to <code>selectSeries</code>)
<code>select.outputs</code>	series to be plotted. (passed to <code>selectSeries</code>)
<code>x.sections</code>	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 <code>x.sections</code> is FALSE.
<code>periods</code>	The periods at which the distribution should be calculated and plotted. The default gives the first three.
<code>graphs.per.page</code>	integer indicating number of graphs to place on a page.
<code>...</code>	(further arguments, currently disregarded).
<code>select</code>	integer vector indicating roots to be plotted. If <code>select</code> is not NULL then roots are sorted by magnitude and only the indicated roots are plotted. For example, <code>select=c(1,2)</code> will plot only the two largest roots.

Details

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

Value

None

See Also

`tfplot.MonteCarloSimulations`

Examples

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

```
distribution.coefEstEval
```

Plot distribution of estimates

Description

Plot distribution of estimates.

Usage

```
## 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

<code>obj</code>	an object as returned by <code>EstEval</code> .
<code>Sort</code>	if <code>Sort</code> is true then sort is applied. This helps (a bit) with estimation methods like <code>black.box</code> which may not return parameters of the same length or in the same order.
<code>bandwidth</code>	passed to <code>density</code> or <code>ksmooth</code> .
<code>graphs.per.page</code>	integer indicating number of graphs to place on a page.

... 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="dse")
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

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

[simulateMonteCarloSimulations](#) [distribution](#) [forecastCovWRTtrue](#)

Examples

```
data("egl.DSE.data.diff", package="dse")
model <- estVARXls(TSdata(output=outputData(egl.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)
```

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

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

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 a normal noise will be generated by `simulate`. This will be iid $N(0,1)$. 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="dse")
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)
```

```
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

<code>model</code>	an object from which a model can be extracted. The model must have an associated simulation method (e.g. a <code>TSmodel</code>).
<code>simulation.args,</code>	A list of arguments in addition to <code>model</code> which are passed to <code>simulate</code> .
<code>replications</code>	The number of simulations.
<code>rng</code>	The RNG and starting seed.
<code>quiet</code>	logical indicating if printing and many warning messages should be suppressed.
<code>obj</code>	an object.
<code>...</code>	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("egl.DSE.data.diff", package="dse")
model <- estVARXls(egl.DSE.data.diff)
z <- MonteCarloSimulations(model, simulation.args=list(sampleT=100))
tfplot(z)
distribution(z)
```

```
nseries.MonteCarloSimulations
      Number of Series
```

Description

Return the number of series.

Usage

```
## 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.

```
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.

```
print.estimatedModels
      Print Specific Methods
```

Description

See the generic function description.

Usage

```
## S3 method for class 'EstEval':
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 <code>digits</code> is NULL then the value of <code>digits</code> specified by <code>options</code> is used.
...	(further arguments, currently disregarded).

See Also

[print summary](#)

roots.coefEstEval *Roots Specific Methods*

Description

See the generic function description.

Usage

```
## 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.
criterion.args	arguments to be passed to this method when it is called by <code>EstEval</code> .
...	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](#)

```
seriesNamesInput.MonteCarloSimulations
      TS Input and Output Specific Methods
```

Description

See the generic function description.

Usage

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

Arguments

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

```
summary.EstEval      Summary Specific Methods
```

Description

See the generic function description.

Usage

```
## 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 '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.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()$digits, ...)
## S3 method for class 'summary.MonteCarloSimulations':
print(x, digits=options()$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 <code>digits</code> is <code>NULL</code> then the value of <code>digits</code> specified by <code>options</code> is used.
series	The series which should be plotted. The default <code>NULL</code> 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.EstEval *Specific Methods for Testing Equality*

Description

See the generic function description.

Usage

```
## S3 method for class 'EstEval':
testEqual(obj1, obj2, fuzz=0)
## 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 <code>fuzz</code> 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 reciprocal 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.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

<code>x</code>	The result of <code>MonteCarloSimulations</code> .
<code>tf</code>	The time frame for plots. see <code>tfplot</code> .
<code>start</code>	The starting period for plots, taken from <code>tf</code> by default.
<code>end</code>	The ending period for plots, taken from <code>tf</code> by default.
<code>series</code>	The series which should be plotted. The default <code>NULL</code> gives all series.
<code>select.simulations</code>	Vector of integers indicating the simulations which should be plotted. The default plots all simulations.
<code>graphs.per.page</code>	The number of graphs to put on a page.
<code>mar</code>	Plot margins (see <code>par</code>).
<code>...</code>	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("egl.DSE.data.diff", package="dse")
model <- estVARXls(egl.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

<code>x</code>	an object for which a tfplot is to be produced.
<code>complex.plane</code>	logical indicating if the plot should be on the complex plane.
<code>cumulate</code>	logical indicating if the cumulative average of roots should be plotted
<code>invert</code>	logical indicating if the inverse of roots should be plotted
<code>Sort</code>	logical indicating if the roots should be sorted.
<code>...</code>	arguments passed to other methods.
<code>norm</code>	logical indicating if the euclidean norm of roots should be plotted (square root of the sum of squared roots).
<code>bounds</code>	logical indicating if estimated one standard error bounds should be plotted around the lines for the true roots.
<code>transform</code>	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 arguments `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 reciprocal 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 '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, ...)
```

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](#)

`TSdata.coefEstEval` *TS Extractor Specific Methods*

Description

See the generic function description.

Usage

```
## S3 method for class 'coefEstEval':
TSestModel(obj)
## S3 method for class 'coefEstEval':
TSmodel(obj, ...)
```

Arguments

`obj` an object from which to extract the `TSmodel` or `TSestModel`.
`...` arguments to be passed to other methods.

See Also

[TSdata](#) [TSestModel](#) [TSmodel](#)

Index

*Topic **programming**

nseries.MonteCarloSimulations,
10
periodsMonteCarloSimulations,
10

*Topic **ts**

coef.TSmodelEstEval, 1
distribution, 2
distribution.coefEstEval, 4
EstEval, 5
generateSSmodel, 6
genMineData, 7
MonteCarloSimulations, 8
nseries.MonteCarloSimulations,
10
periodsMonteCarloSimulations,
10
print.estimatedModels, 11
roots.coefEstEval, 12
seriesNamesInput.MonteCarloSimulations,
12
summary.EstEval, 13
testEqual.EstEval, 14
tfplot.coefEstEval, 15
tfplot.MonteCarloSimulations,
16
tfplot.rootsEstEval, 17
tfplot.TSdata.ee, 18
TSdata.coefEstEval, 19

*Topic **utilities**

nseries.MonteCarloSimulations,
10
periodsMonteCarloSimulations,
10

build.diagonal.model
(genMineData), 7

build.input.models (genMineData),
7

coef, 2

coef.TSestModelEstEval
(coef.TSmodelEstEval), 1
coef.TSmodelEstEval, 1

distribution, 2, 6, 9
distribution.coefEstEval, 4
distribution.MonteCarloSimulations,
16
distribution.rootsEstEval
(distribution.coefEstEval),
4

EstEval, 2, 4, 5, 9, 12, 15, 18, 19

forecastCovWRTtrue, 6, 9

generateSSmodel, 6
genMineData, 7

is.EstEval (EstEval), 5
is.MonteCarloSimulations
(MonteCarloSimulations), 8

MonteCarloSimulations, 6, 8

nseries.MonteCarloSimulations, 10
nseriesInput.MonteCarloSimulations
(nseries.MonteCarloSimulations),
10

nseriesOutput.MonteCarloSimulations
(nseries.MonteCarloSimulations),
10

periods.MonteCarloSimulations
(periodsMonteCarloSimulations),
10

periodsMonteCarloSimulations, 10

plot.rootsEstEval
(tfplot.rootsEstEval), 17

print, 11, 14

```

print.EstEval
    (print.estimatedModels), 11
print.estimatedModels, 11
print.MonteCarloSimulations
    (print.estimatedModels), 11
print.summary.coefEstEval
    (summary.EstEval), 13
print.summary.EstEval
    (summary.EstEval), 13
print.summary.MonteCarloSimulations
    (summary.EstEval), 13
print.summary.rootsEstEval
    (summary.EstEval), 13
print.summary.TSestModelEstEval
    (summary.EstEval), 13
print.summary.TSmodelEstEval
    (summary.EstEval), 13

roots, 12
roots.coefEstEval, 12
roots.rootsEstEval
    (roots.coefEstEval), 12
roots.TSestModelEstEval
    (roots.coefEstEval), 12
roots.TSmodelEstEval
    (roots.coefEstEval), 12

seriesNamesInput.MonteCarloSimulations,
    12
seriesNamesOutput.MonteCarloSimulations
    (seriesNamesInput.MonteCarloSimulations),
    12
simulate, 6, 8, 9
stability, 12
summary, 11, 14
summary.coefEstEval
    (summary.EstEval), 13
summary.EstEval, 13
summary.MonteCarloSimulations
    (summary.EstEval), 13
summary.rootsEstEval
    (summary.EstEval), 13
summary.TSestModelEstEval
    (summary.EstEval), 13
summary.TSmodelEstEval
    (summary.EstEval), 13

testEqual, 14
testEqual.EstEval, 14
testEqual.MonteCarloSimulations
    (testEqual.EstEval), 14
tfplot, 15, 18, 19
tfplot.coefEstEval, 15
tfplot.EstEval
    (tfplot.TSdata.ee), 18
tfplot.MonteCarloSimulations, 3,
    16
tfplot.rootsEstEval, 17
tfplot.TSdata.ee, 18
tfplot.TSestModelEstEval
    (tfplot.TSdata.ee), 18
tfplot.TSmodelEstEval
    (tfplot.TSdata.ee), 18
tframe.MonteCarloSimulations
    (periodsMonteCarloSimulations),
    10
TSdata, 19
TSdata.coefEstEval, 19
TSestModel, 19
TSestModel.coefEstEval
    (TSdata.coefEstEval), 19
TSmodel, 19
TSmodel.coefEstEval
    (TSdata.coefEstEval), 19

```