Package 'greybox'

May 25, 2018

Type Package

Title Toolbox for Model Building and Forecasting

Version 0.2.2

Date 2018-05-25

URL https://github.com/config-i1/greybox

BugReports https://github.com/config-i1/greybox/issues

Description Implements functions and instruments for regression model building and its application to forecasting. The main scope of the package is in variables selection and models specification for cases of time series data. This includes promotional modelling, selection between different dynamic regressions with non-standard distributions of errors, selection based on cross validation, solutions to the fat regressions model problem and more. Models developed in the package are tailored specifically for forecasting purposes. So as a results there are several methods that allow producing forecasts from these models and visualising them.

License GPL (>= 2)

Depends R (>= 3.0.2)

Imports forecast, stats, graphics, utils, lamW

Suggests smooth, doMC, doParallel, foreach, numDeriv, testthat, rmarkdown, knitr

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Ivan Svetunkov [aut, cre] (Lecturer at Centre for Marketing Analytics and Forecasting, Lancaster University, UK)

Maintainer Ivan Svetunkov <ivan@svetunkov.ru>

Repository CRAN

Date/Publication 2018-05-25 07:32:53 UTC

R topics documented:

`AICc` *Corrected Akaike's Information Criterion and Bayesian Information Criterion*

Description

This function extracts AICc / BICc from models. It can be applied to wide variety of models that use logLik() and nobs() methods (including the popular lm, forecast, smooth classes).

Usage

```
AICc(object, ...)

BICc(object, ...)
```

Arguments

`object`	Time series model.
`...`	Some stuff.

Details

AICc was proposed by Nariaki Sugiura in 1978 and is used on small samples for the models with normally distributed residuals. BICc was derived in McQuarrie (1999) and is used in similar circumstances.

IMPORTANT NOTE: both of the criteria can only be used for univariate models (regression models, ARIMA, ETS etc) with normally distributed residuals!

Value

This function returns numeric value.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Burnham Kenneth P. and Anderson David R. (2002). Model Selection and Multimodel Inference. A Practical Information-Theoretic Approach. Springer-Verlag New York. DOI: [10.1007/b97636](http://dx.doi.org/10.1007/b97636).
- McQuarrie A.D., A small-sample correction for the Schwarz SIC model selection criterion, Statistics & Probability Letters 44 (1999) pp.79-86. doi: 10.1016/S01677152(98)002946
- Sugiura Nariaki (1978) Further analysts of the data by Akaike's information criterion and the finite corrections, Communications in Statistics - Theory and Methods, 7:1, 13-26, doi: 10.1080/03610927808827599

See Also

AIC, BIC

Examples

```
xreg <- cbind(rnorm(100,10,3),rnorm(100,50,5))
xreg <- cbind(100+0.5*xreg[,1]-0.75*xreg[,2]+rnorm(100,0,3),xreg,rnorm(100,300,10))
colnames(xreg) <- c("y","x1","x2","Noise")

ourModel <- stepwise(xreg)

AICc(ourModel,h=10)
BICc(ourModel,h=10)
```

`determination` *Determination coefficients*

Description

Function produces determination coefficient for the provided data

Usage

```
determination(xreg, ...)
```

Arguments

`xreg`	Data frame or a matrix, containing the exogenous variables.
`...`	Other values passed to cor function.

Details

The function calculates determination coefficients (aka R^2) between all the provided variables. The higher the coefficient is, the higher the potential multicollinearity effect in the model with the variables will be. Coefficients of determination are connected directly to Variance Inflation Factor (VIF): VIF = 1 / (1 - determination). Arguably it is easier to interpret, because it is restricted with (0, 1) bounds. The multicollinearity can be considered as serious, when determination > 0.9 (which corresponds to VIF > 10).

Value

Function returns the vector of determination coefficients.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

cor

Examples

```
### Simple example
xreg <- cbind(rnorm(100,10,3),rnorm(100,50,5))
xreg <- cbind(100+0.5*xreg[,1]-0.75*xreg[,2]+rnorm(100,0,3),xreg,rnorm(100,300,10))
colnames(xreg) <- c("x1","x2","x3","Noise")
determination(xreg)
```

---

dlaplace *Laplace Distribution*

Description

Density, cumulative distribution, quantile functions and random generation for the Laplace distribution with the location parameter mu and Mean Absolute Error (or Mean Absolute Deviation) equal to b.

Usage

```
dlaplace(q, mu = 0, b = 1, log = FALSE)

plaplace(q, mu = 0, b = 1)

qlaplace(p, mu = 0, b = 1)

rlaplace(n = 1, mu = 0, b = 1)
```

Arguments

q	vector of quantiles.
mu	vector of location parameters (means).
b	vector of mean absolute errors.
log	if TRUE, then probabilities are returned in logarithms.
p	vector of probabilities.
n	number of observations. Should be a single number.

Details

When mu=0 and b=1, the Laplace distribution becomes standardized with. The distribution has the following density function:

f(x) = 1/(2b) exp(-abs(x-mu) / b)

Value

Depending on the function, various things are returned (usually either vector or scalar):

- ds returns the density function value for the provided parameters.
- ps returns the value of the cumulative function for the provided parameters.
- qs returns quantiles of the distribution. Depending on what was provided in p, mu and b, this can be either a vector or a matrix, or an array.
- rs returns a vector of random variables generated from the Laplace distribution. Depending on what was provided in mu and b, this can be either a vector or a matrix or an array.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Wikipedia page on Laplace distribution: https://en.wikipedia.org/wiki/Laplace_distribution.

Examples

```
x <- dlaplace(c(-1000:1000)/10, 0, 1)
plot(x, type="l")

x <- plaplace(c(-1000:1000)/10, 0, 1)
plot(x, type="l")

qlaplace(c(0.025,0.975), 0, c(1,2))

x <- rlaplace(1000, 0, 1)
hist(x)
```

ds *S Distribution*

Description

Density, cumulative distribution, quantile functions and random generation for the S distribution with the location parameter mu and a scaling parameter b.

Usage

```
ds(q, mu = 0, b = 1, log = FALSE)

ps(q, mu = 0, b = 1)

qs(p, mu = 0, b = 1)

rs(n = 1, mu = 0, b = 1)
```

Arguments

q	vector of quantiles.
mu	vector of location parameters (means).
b	vector of scaling parameter (which equals to ham/2).
log	if TRUE, then probabilities are returned in logarithms.
p	vector of probabilities.
n	number of observations. Should be a single number.

Details

When mu=0 and ham=2, the S distribution becomes standardized with b=1 (this is because b=ham/2). The distribution has the following density function:

f(x) = 1/(4b^2) exp(-sqrt(abs(x-mu)) / b)

The S distribution has fat tails and large excess.

Value

Depending on the function, various things are returned (usually either vector or scalar):

- ds returns the density function value for the provided parameters.
- ps returns the value of the cumulative function for the provided parameters.
- qs returns quantiles of the distribution. Depending on what was provided in p, mu and b, this can be either a vector or a matrix, or an array.
- rs returns a vector of random variables generated from the S distribution. Depending on what was provided in mu and b, this can be either a vector or a matrix or an array.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

Examples

```
x <- ds(c(-1000:1000)/10, 0, 1)
plot(x, type="l")

x <- ps(c(-1000:1000)/10, 0, 1)
plot(x, type="l")

qs(c(0.025,0.975), 0, 1)

x <- rs(1000, 0, 1)
hist(x)
```

---

greybox *Grey box*

Description

Toolbox for working with multivariate models for purposes of analysis and forecasting

Details

Package:	greybox
Type:	Package
Date:	2018-02-13 - Inf
License:	GPL-2

The following functions are included in the package:

- AICc and BICc - AIC / BIC corrected for the sample size.
- determination - Coefficients of determination between different exogenous variables.
- stepwise - Stepwise based on information criteria and partial correlations. Efficient and fast.
- xregExpander - Function that expands the provided data into the data with lags and leads.
- lmCombine - Function combines lm models from the estimated based on information criteria weights.
- ro - Rolling origin evaluation.
- qlaplace, dlaplace, plaplace, rlaplace - Laplace distribution and the respective functions.
- qs, ds, ps, rs - S distribution and the respective functions.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

Maintainer: Ivan Svetunkov

See Also

stepwise, lmCombine

Examples

```
## Not run:
xreg <- cbind(rnorm(100,10,3),rnorm(100,50,5))
xreg <- cbind(100+0.5*xreg[,1]-0.75*xreg[,2]+rnorm(100,0,3),xreg,rnorm(100,300,10))
colnames(xreg) <- c("y","x1","x2","Noise")

stepwise(xreg)

## End(Not run)
```

---

lmCombine *Combine regressions based on information criteria*

Description

Function combines parameters of linear regressions of the first variable on all the other provided data.

Usage

```
lmCombine(data, ic = c("AICc", "AIC", "BIC", "BICc"), bruteForce = FALSE,
  silent = TRUE)
```

Arguments

data	Data frame containing dependent variable in the first column and the others in the rest.
ic	Information criterion to use.
bruteForce	If TRUE, then all the possible models are generated and combined. Otherwise the best model is found and then models around that one are produced and then combined.
silent	If FALSE, then nothing is silent, everything is printed out. TRUE means that nothing is produced.

Details

The algorithm uses lm() to fit different models and then combines the models based on the selected IC.

Value

Function returns `model` - the final model of the class "lm.combined".

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Burnham Kenneth P. and Anderson David R. (2002). Model Selection and Multimodel Inference. A Practical Information-Theoretic Approach. Springer-Verlag New York. DOI: [10.1007/b97636](http://dx.doi.org/10.1007/b97636).

See Also

`step`, `xregExpander`,`stepwise`

Examples

```
### Simple example
xreg <- cbind(rnorm(100,10,3),rnorm(100,50,5))
xreg <- cbind(100+0.5*xreg[,1]-0.75*xreg[,2]+rnorm(100,0,3),xreg,rnorm(100,300,10))
colnames(xreg) <- c("y","x1","x2","Noise")
inSample <- xreg[1:80,]
outSample <- xreg[-c(1:80),]
# Combine all the possible models
ourModel <- lmCombine(inSample,bruteForce=TRUE)
forecast(ourModel,outSample)
plot(forecast(ourModel,outSample))

### Fat regression example
xreg <- matrix(rnorm(5000,10,3),50,100)
xreg <- cbind(100+0.5*xreg[,1]-0.75*xreg[,2]+rnorm(50,0,3),xreg,rnorm(50,300,10))
colnames(xreg) <- c("y",paste0("x",c(1:100)),"Noise")
inSample <- xreg[1:40,]
outSample <- xreg[-c(1:40),]
# Combine only the models close to the optimal
ourModel <- lmCombine(inSample,ic="BICc",bruteForce=FALSE)
summary(ourModel)
plot(forecast(ourModel,outSample))
```

nParam *Number of parameters in the model*

Description

This function returns the number of estimated parameters in the model

Usage

```
nParam(object, ...)
```

Arguments

object	Time series model.
...	Some other parameters passed to the method.

Details

This is a very basic and a simple function which does what it says: extracts number of parameters in the estimated model.

Value

This function returns a numeric value.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

nobs, logLik

Examples

```
### Simple example
xreg <- cbind(rnorm(100,10,3),rnorm(100,50,5))
xreg <- cbind(100+0.5*xreg[,1]-0.75*xreg[,2]+rnorm(100,0,3),xreg,rnorm(100,300,10))
colnames(xreg) <- c("y","x1","x2","Noise")
ourModel <- lm(y~.,data=as.data.frame(xreg))

nParam(ourModel)
```

ro *Rolling Origin*

Description

The function does rolling origin for any forecasting function

Usage

```
ro(data, h = 10, origins = 10, call, value = NULL, ci = FALSE,
  co = FALSE, silent = TRUE, parallel = FALSE)
```

Arguments

data	Data vector or ts object passed to the function.
h	The forecasting horizon.
origins	The number of rolling origins.
call	The call that is passed to the function. The call must be in quotes. Example: "forecast(ets(data),h)". Here data shows where the data is and h defines where the horizon should be passed in the call. Some hidden parameters can also be specified in the call. For example, parameters counti, counto and countf are used in the inner loop and can be used for the regulation of exogenous variables sizes. See examples for the details.
value	The variable or set of variables returned by the call. For example, mean for functions of forecast package. This can also be a vector of variables. See examples for the details. If the parameter is NULL, then all the values from the call are returned (could be really messy!). Note that if your function returns a list with matrices, then ro will return an array. If your function returns a list, then you will have a list of lists in the end. So it makes sense to understand what you want to get before running the function.
ci	The parameter defines if the in-sample window size should be constant. If TRUE, then with each origin one observation is added at the end of series and another one is removed from the beginning.
co	The parameter defines whether the holdout sample window size should be constant. If TRUE, the rolling origin will stop when less than h observations are left in the holdout.
silent	If TRUE, nothing is printed out in the console.
parallel	If TRUE, then the rolling origin is done in parallel. WARNING! Packages foreach and either doMC (Linux only), doParallel or doSNOW are needed in order to run the function in parallel.

Details

This function produces rolling origin forecasts using the `data` and a `call` passed as parameters. The function can do all of that either in serial or in parallel, but it needs `foreach` and either `doMC` (Linux only), `doParallel` or `doSNOW` packages installed in order to do the latter.

This is a dangerous function, so be careful with the call that you pass to it, and make sure that it is well formulated before the execution.

Value

Function returns the following variables:

- `actuals` - the data provided to the function.
- `holdout` - the matrix of actual values corresponding to the produced forecasts from each origin.
- `value` - the matrices / array / lists with the produced data from each origin. Name of each object corresponds to the names in the parameter `value`.

Author(s)

Yves Sagaert

Ivan Svetunkov, `<ivan@svetunkov.ru>`

References

- Tashman, (2000) Out-of-sample tests of forecasting accuracy: an analysis and review International Journal of Forecasting, 16, pp. 437-450. https://doi.org/10.1016/S0169-2070(00)00065-0.

Examples

```
x <- rnorm(100,0,1)
ourCall <- "predict(arima(x=data,order=c(0,1,1)),n.ahead=h)"

# The default call and values
ourValue <- "pred"
ourRO <- ro(x, h=5, origins=5, ourCall, ourValue)

# We can now plot the results of this evaluation:
plot(ourRO)

# You can also use dolar sign
ourValue <- "$pred"
# And you can have constant in-sample size
ro(x, h=5, origins=5, ourCall, ourValue, ci=TRUE)

# You can ask for several values
ourValue <- c("pred","se")
# And you can have constant holdout size
ro(x, h=5, origins=20, ourCall, ourValue, ci=TRUE, co=TRUE)
```

```
#### The following code will give exactly the same result as above,
#### but computed in parallel using all but 1 core of CPU:
## Not run: ro(x, h=5, origins=20, ourCall, ourValue, ci=TRUE, co=TRUE, parallel=TRUE)

#### If you want to use functions from forecast package, please note that you need to
#### set the values that need to be returned explicitly. There are two options for this.
# Example 1:
## Not run: ourCall <- "forecast(ets(data), h=h, level=95)"
ourValue <- c("mean", "lower", "upper")
ro(x,h=5,origins=5,ourCall,ourValue)
## End(Not run)

# Example 2:
## Not run: ourCall <- "forecast(ets(data), h=h, level=c(80,95))"
ourValue <- c("mean", "lower[,1]", "upper[,1]", "lower[,2]", "upper[,2]")
ro(x,h=5,origins=5,ourCall,ourValue)
## End(Not run)

#### A more complicated example using the for loop and
#### several time series
x <- matrix(rnorm(120*3,0,1), 120, 3)

## Form an array for the forecasts we will produce
## We will have 4 origins with 6-steps ahead forecasts
ourForecasts <- array(NA,c(6,4,3))

## Define models that need to be used for each series
ourModels <- list(c(0,1,1), c(0,0,1), c(0,1,0))

## This call uses specific models for each time series
ourCall <- "predict(arima(data, order=ourModels[[i]]), n.ahead=h)"
ourValue <- "pred"

## Start the loop. The important thing here is to use the same variable 'i' as in ourCall.
for(i in 1:3){
    ourdata <- x[,i]
    ourForecasts[,,i] <- ro(data=ourdata,h=6,origins=4,call=ourCall,
                            value=ourValue,co=TRUE,silent=TRUE)$pred
}

## ourForecasts array now contains rolling origin forecasts from specific
## models.

##### An example with exogenous variables
x <- rnorm(100,0,1)
xreg <- rnorm(100,0,1)

## 'counti' is used to define in-sample size of xreg,
## 'counto' - the size of the holdout sample of xreg

ourCall <- "predict(arima(x=data, order=c(0,1,1), xreg=xreg[counti]),
            n.ahead=h, newxreg=xreg[counto])"
```

```
ourValue <- "pred"
ro(x,h=5,origins=5,ourCall,ourValue)


## 'countf' is used to take xreg of the size corresponding to the whole
## sample on each iteration
## This is useful when working with functions from smooth package.
## The following call will return the forecasts from es() function of smooth.
## Not run: ourCall <- "es(data=data, h=h, xreg=xreg[countf])"
ourValue <- "forecast"
ro(x,h=5,origins=5,ourCall,ourValue)
## End(Not run)
```

`stepwise` *Stepwise selection of regressors*

Description

Function selects variables that give linear regression with the lowest information criteria. The selection is done stepwise (forward) based on partial correlations. This should be a simpler and faster implementation than step() function from 'stats' package.

Usage

```
stepwise(data, ic = c("AICc", "AIC", "BIC", "BICc"), silent = TRUE,
  df = NULL, method = c("pearson", "kendall", "spearman"))
```

Arguments

`data`	Data frame containing dependant variable in the first column and the others in the rest.
`ic`	Information criterion to use.
`silent`	If `silent=FALSE`, then nothing is silent, everything is printed out. `silent=TRUE` means that nothing is produced.
`df`	Number of degrees of freedom to add (should be used if stepwise is used on residuals).
`method`	Method of correlations calculation. The default is Kendall's Tau, which should be applicable to a wide range of data in different scales.

Details

The algorithm uses lm() to fit different models and cor() to select the next regressor in the sequence.

Value

Function returns `model` - the final model of the class "lm".

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Burnham Kenneth P. and Anderson David R. (2002). Model Selection and Multimodel Inference. A Practical Information-Theoretic Approach. Springer-Verlag New York. DOI: [10.1007/b97636](http://dx.doi.org/10.1007/b97636).

See Also

step, xregExpander,combine

Examples

```
### Simple example
xreg <- cbind(rnorm(100,10,3),rnorm(100,50,5))
xreg <- cbind(100+0.5*xreg[,1]-0.75*xreg[,2]+rnorm(100,0,3),xreg,rnorm(100,300,10))
colnames(xreg) <- c("y","x1","x2","Noise")
stepwise(xreg)

### Fat regression example
xreg <- matrix(rnorm(20000,10,3),100,200)
xreg <- cbind(100+0.5*xreg[,1]-0.75*xreg[,2]+rnorm(100,0,3),xreg,rnorm(100,300,10))
colnames(xreg) <- c("y",paste0("x",c(1:200)),"Noise")
ourModel <- stepwise(xreg,ic="AICc")
plot(ourModel$ICs,type="l",ylim=range(min(ourModel$ICs),max(ourModel$ICs)+5))
points(ourModel$ICs)
text(c(1:length(ourModel$ICs))+0.1,ourModel$ICs+5,names(ourModel$ICs))
```

---

xregExpander *Exogenous variables expander*

Description

Function expands the provided matrix or vector of variables, producing values with lags and leads specified by lags variable.

Usage

```
xregExpander(xreg, lags = c(-frequency(xreg):frequency(xreg)),
  silent = TRUE)
```

Arguments

xreg	Vector / matrix / data.frame, containing variables that need to be expanded. In case of vector / matrix it is recommended to provide ts object, so the frequency of the data is taken into account.
lags	Vector of lags / leads that we need to have. Negative values mean lags, positive ones mean leads.
silent	If silent=FALSE, then the progress is printed out. Otherwise the function won't print anything in the console.

Details

This function could be handy when you want to check if lags and leads of a variable influence the dependent variable. Can be used together with xregDo="select" in es, ces, ges and ssarima. All the missing values in the beginning and at the end of lagged series are substituted by mean forecasts produced using es.

Value

ts matrix with the expanded variables is returned.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

es, stepwise

Examples

```
# Create matrix of two variables, make it ts object and expand it
x <- cbind(rnorm(100,100,1),rnorm(100,50,3))
x <- ts(x,frequency=12)
xregExpander(x)
```

Index