Package 'greybox'

July 31, 2019

Type Package

Title Toolbox for Model Building and Forecasting

Version 0.5.3

Date 2019-07-31

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

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

Language en-GB

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 regression 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, numDeriv, nloptr

LinkingTo Rcpp

Suggests smooth (>= 2.5.1), doMC, doParallel, foreach, testthat, rmarkdown, knitr

Enhances vars

RoxygenNote 6.1.1

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation yes

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 2019-07-31 13:40:02 UTC

R topics documented:

actuals *Function extracts the actual values from the function*

Description

This is a simple method that returns the values of the response variable of the model

Usage

```
actuals(object, ...)

## Default S3 method:
actuals(object, ...)

## S3 method for class 'alm'
actuals(object, ...)
```

Arguments

object	Model estimated using one of the functions of smooth package.
...	A parameter all can also be provided here. If it is FALSE, then in the case of occurrence model, only demand sizes will be returned. Works only with 'alm' class.

Value

The vector of the response variable.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

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)

actuals(ourModel)
```

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)
BICc(ourModel)
```

`alm` *Advanced Linear Model*

Description

Function estimates model based on the selected distribution

Usage

```
alm(formula, data, subset, na.action, distribution = c("dnorm", "dlogis",
  "dlaplace", "dalaplace", "ds", "dt", "dfnorm", "dlnorm", "dchisq",
  "dbcnorm", "dpois", "dnbinom", "dbeta", "plogis", "pnorm"),
  occurrence = c("none", "plogis", "pnorm"), ar = 0, i = 0,
  parameters = NULL, vcovProduce = FALSE, fast = TRUE, ...)
```

Arguments

`formula`	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted.
`data`	a data frame or a matrix, containing the variables in the model.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The factory-fresh default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
`distribution`	what density function to use in the process. The full name of the distribution should be provided here. Values with "d" in the beginning of the name refer to the density function, while "p" stands for "probability" (cumulative distribution function). The names align with the names of distribution functions in R. For example, see dnorm.

`occurrence` what distribution to use for occurrence variable. Can be `"none"`, then nothing happens; `"plogis"` - then the logistic regression using `alm()` is estimated for the occurrence part; `"pnorm"` - then probit is constructed via `alm()` for the occurrence part. In both of the latter cases, the formula used is the same as the formula for the sizes. Finally, an "alm" model can be provided and its estimates will be used in the model construction.

If this is not `"none"`, then the model is estimated in two steps: 1. Occurrence part of the model; 2. Sizes part of the model (excluding zeroes from the data).

`ar` the order of AR to include in the model. Only non-seasonal orders are accepted.

`i` the order of I to include in the model. Only non-seasonal orders are accepted.

`parameters` vector of parameters of the linear model. When `NULL`, it is estimated.

`vcovProduce` whether to produce variance-covariance matrix of coefficients or not. This is done via hessian calculation, so might be computationally costly.

`fast` if `FALSE`, then the function won't check whether the data has variability and whether the regressors are correlated. Might cause trouble, especially in cases of multicollinearity.

`...` additional parameters to pass to distribution functions. This includes: `alpha` value for Asymmetric Laplace distribution, `size` for the Negative Binomial or `df` for the Chi-Squared. You can also pass two parameters to the optimiser: 1. `maxeval` - maximum number of evaluations to carry out (default is 100); 2. `xtol_rel` - the precision of the optimiser (the default is 1E-6); 3. `algorithm` - the algorithm to use in optimisation (`"NLOPT_LN_SBPLX"` by default). 4. `print_level` - the level of output for the optimiser (0 by default). You can read more about these parameters in the documentation of nloptr function.

Details

This is a function, similar to lm, but for the cases of several non-normal distributions. These include:

1. Normal distribution, dnorm,
2. Logistic Distribution, dlogis,
3. Laplace distribution, dlaplace,
4. Asymmetric Laplace distribution, dalaplace,
5. T-distribution, dt,
6. S-distribution, ds,
7. Folded normal distribution, dfnorm,
8. Log normal distribution, dlnorm,
9. Chi-Squared Distribution, dchisq,
10. Beta distribution, dbeta,
11. Poisson Distribution, dpois,
12. Negative Binomial Distribution, dnbinom,
13. Cumulative Logistic Distribution, plogis,
14. Cumulative Normal distribution, pnorm.

This function is slower than `lm`, because it relies on likelihood estimation of parameters, hessian calculation and matrix multiplication. So think twice when using `distribution="dnorm"` here.

Probably some other distributions will be added to this function at some point...

The estimation is done using likelihood of respective distributions.

ALM function currently does not work with factors and does not accept transformations of variables in the formula. So you need to do transformations separately before using the function.

See more details and examples in the vignette "ALM": `vignette("alm","greybox")`

Value

Function returns `model` - the final model of the class "alm", which contains:

- coefficients - estimated parameters of the model,
- vcov - covariance matrix of parameters of the model (based on Fisher Information). Returned only when `vcovProduce=TRUE`,
- fitted - fitted values,
- residuals - residuals of the model,
- mu - the estimated location parameter of the distribution,
- scale - the estimated scale parameter of the distribution,
- distribution - distribution used in the estimation,
- logLik - log-likelihood of the model,
- df.residual - number of degrees of freedom of the residuals of the model,
- df - number of degrees of freedom of the model,
- call - how the model was called,
- rank - rank of the model,
- data - data used for the model construction,
- occurrence - the occurrence model used in the estimation,
- other - the list of all the other parameters either passed to the function or estimated in the process, but not included in the standard output (e.g. `alpha` for Asymmetric Laplace).

Author(s)

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

See Also

`stepwise`,`lmCombine`,`xregTransformer`

Examples

```
### An example with mtcars data and factors
mtcars2 <- within(mtcars, {
   vs <- factor(vs, labels = c("V", "S"))
   am <- factor(am, labels = c("automatic", "manual"))
   cyl  <- ordered(cyl)
   gear <- ordered(gear)
   carb <- ordered(carb)
})
# The standard model with Log Normal distribution
ourModel <- alm(mpg~., mtcars2[1:30,], distribution="dlnorm")
summary(ourModel)
plot(ourModel)

# Produce predictions with the one sided interval (upper bound)
predict(ourModel, mtcars2[-c(1:30),], interval="p", side="u")


### Artificial data for the other examples
xreg <- cbind(rlaplace(100,10,3),rnorm(100,50,5))
xreg <- cbind(100+0.5*xreg[,1]-0.75*xreg[,2]+rlaplace(100,0,3),xreg,rnorm(100,300,10))
colnames(xreg) <- c("y","x1","x2","Noise")
inSample <- xreg[1:80,]
outSample <- xreg[-c(1:80),]

# An example with Laplace distribution
ourModel <- alm(y~x1+x2, inSample, distribution="dlaplace")
summary(ourModel)
plot(predict(ourModel,outSample))

# And another one with Asymmetric Laplace distribution (quantile regression)
# with optimised alpha
ourModel <- alm(y~x1+x2, inSample, distribution="dalaplace")
summary(ourModel)
plot(predict(ourModel,outSample))

# An example with AR(1) order
ourModel <- alm(y~x1+x2, inSample, distribution="dnorm", ar=1)
summary(ourModel)
plot(predict(ourModel,outSample))

### Examples with the count data
xreg[,1] <- round(exp(xreg[,1]-70),0)
inSample <- xreg[1:80,]
outSample <- xreg[-c(1:80),]

# Negative Binomial distribution
ourModel <- alm(y~x1+x2, inSample, distribution="dnbinom")
summary(ourModel)
predict(ourModel,outSample,interval="p",side="u")
```

```
# Poisson distribution
ourModel <- alm(y~x1+x2, inSample, distribution="dpois")
summary(ourModel)
predict(ourModel,outSample,interval="p",side="u")


### Examples with binary response variable
xreg[,1] <- round(xreg[,1] / (1 + xreg[,1]),0)
inSample <- xreg[1:80,]
outSample <- xreg[-c(1:80),]

# Logistic distribution (logit regression)
ourModel <- alm(y~x1+x2, inSample, distribution="plogis")
summary(ourModel)
plot(predict(ourModel,outSample,interval="c"))

# Normal distribution (probit regression)
ourModel <- alm(y~x1+x2, inSample, distribution="pnorm")
summary(ourModel)
plot(predict(ourModel,outSample,interval="p"))
```

`association` *Measures of association*

Description

Function returns the matrix of measures of association for different types of variables.

Usage

```
association(x, y = NULL, use = c("na.or.complete", "complete.obs",
  "everything", "all.obs"))

assoc(x, y = NULL, use = c("na.or.complete", "complete.obs",
  "everything", "all.obs"))
```

Arguments

`x`	Either data.frame or a matrix
`y`	The numerical variable.
`use`	What observations to use. See cor function for details. The only option that is not available here is `"pairwise.complete.obs"`.

Details

The function looks at the types of the variables and calculates different measures depending on the result:

- If both variables are numeric, then Pearson's correlation is calculated;
- If both variables are categorical, then Cramer's V is calculated;
- Finally, if one of the variables is categorical, and the other is numeric, then multiple correlation is returned.

After that the measures are wrapped up in a matrix.

Function also calculates the p-values associated with the respective measures (see the return).

See details in the vignette "Marketing analytics with greybox": `vignette("maUsingGreybox","greybox")`

`assoc()` is just a short name for the `association{}`.

Value

The following list of values is returned:

- value - Matrix of the coefficients of association;
- p.value - The p-values for the parameters;
- type - The matrix of the types of measures of association.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

`table`,`tableplot`,`spread`,`cramer`,`mcor`

Examples

```
association(mtcars)
```

`cramer` *Calculate Cramer's V for categorical variables*

Description

Function calculates Cramer's V for two categorical variables based on the table function

Usage

```
cramer(x, y, use = c("na.or.complete", "complete.obs", "everything",
  "all.obs"))
```

Arguments

x	First categorical variable.
y	Second categorical variable.
use	What observations to use. See cor function for details. The only option that is not available here is "pairwise.complete.obs".

Details

The function calculates Cramer's V and also returns the associated statistics from Chi-Squared test with the null hypothesis of independence of the two variables.

See details in the vignette "Marketing analytics with greybox": `vignette("maUsingGreybox","greybox")`

Value

The following list of values is returned:

- valueThe value of Cramer's V;
- statisticThe value of Chi squared statistic associated with the Cramer's V;
- p.valueThe p-value of Chi squared test associated with the Cramer's V;
- dfThe number of degrees of freedom from the test.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

table, tableplot, spread, mcor, association

Examples

```
cramer(mtcars$am, mtcars$gear)
```

---

dalaplace *Asymmetric Laplace Distribution*

Description

Density, cumulative distribution, quantile functions and random number generation for the Asymmetric Laplace distribution with the location parameter mu, Mean Absolute Error (or Mean Absolute Deviation) equal to 2 scale and asymmetry parameter alpha.

Usage

```
dalaplace(q, mu = 0, scale = 1, alpha = 0.5, log = FALSE)

palaplace(q, mu = 0, scale = 1, alpha = 0.5)

qalaplace(p, mu = 0, scale = 1, alpha = 0.5)

ralaplace(n = 1, mu = 0, scale = 1, alpha = 0.5)
```

Arguments

q	vector of quantiles.
mu	vector of location parameters (means).
scale	vector of scale parameters.
alpha	value of asymmetry parameter. Varies from 0 to 1.
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 scale=1, the Laplace distribution becomes standardized. The distribution has the following density function:

f(x) = alpha (1-alpha) / scale exp(-(x-mu)/scale (alpha - I(x<=mu))),

where I(.) is the indicator function (equal to 1 if the condition is satisfied and zero otherwise).

When alpha=0.5, then the distribution becomes Symmetric Laplace, where scale = 1/2 MAE.

This distribution function aligns with the quantile estimates of parameters (Geraci & Bottai, 2007).

Finally, both palaplace and qalaplace are returned for the lower tail of the distribution.

Value

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

- dalaplace returns the density function value for the provided parameters.
- palaplace returns the value of the cumulative function for the provided parameters.
- qalaplace returns quantiles of the distribution. Depending on what was provided in p, mu and scale, this can be either a vector or a matrix, or an array.
- ralaplace returns a vector of random variables generated from the Laplace distribution. Depending on what was provided in mu and scale, this can be either a vector or a matrix or an array.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Geraci Marco, Bottai Matteo (2007). Quantile regression for longitudinal data using the asymmetric Laplace distribution. Biostatistics (2007), 8, 1, pp. 140-154 https://doi.org/10.1093/biostatistics/kxj039
- Yu, K., & Zhang, J. (2005). A three-parameter asymmetric laplace distribution and its extension. Communications in Statistics - Theory and Methods, 34, 1867-1879. https://doi.org/10.1080/03610920500199018

Examples

```
x <- dalaplace(c(-100:100)/10, 0, 1, 0.2)
plot(x, type="l")

x <- palaplace(c(-100:100)/10, 0, 1, 0.2)
plot(x, type="l")

qalaplace(c(0.025,0.975), 0, c(1,2), c(0.2,0.3))

x <- ralaplace(1000, 0, 1, 0.2)
hist(x)
```

---

dbcnorm *Box-Cox Normal Distribution*

Description

Density, cumulative distribution, quantile functions and random number generation for the distribution that becomes normal after the Box-Cox transformation. Note that this is based on the original Box-Cox paper.

Usage

```
dbcnorm(q, mu = 0, sigma = 1, lambda = 0, log = FALSE)

pbcnorm(q, mu = 0, sigma = 1, lambda = 0)

qbcnorm(p, mu = 0, sigma = 1, lambda = 0)

rbcnorm(n = 1, mu = 0, sigma = 1, lambda = 0)
```

Arguments

q	vector of quantiles.
mu	vector of location parameters (means).
sigma	vector of scale parameters.
lambda	tha value of the Box-Cox transform parameter.

log	if TRUE, then probabilities are returned in logarithms.
p	vector of probabilities.
n	number of observations. Should be a single number.

Details

The distribution has the following density function:

f(x) = x^lambda-1 1/sqrt(2 pi) exp(-((y^lambda-1)/lambda -mu)^2 / (2 sigma^2))

Both pbcnorm and qbcnorm are returned for the lower tail of the distribution.

In case of lambda=0, the values of the log normal distribution are returned. In case of lambda=1, the values of the normal distribution are returned with mu=mu+1.

All the functions are defined for non-negative values only.

Value

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

- dbcnorm returns the density function value for the provided parameters.
- pbcnorm returns the value of the cumulative function for the provided parameters.
- qbcnorm returns quantiles of the distribution. Depending on what was provided in p, mu and sigma, this can be either a vector or a matrix, or an array.
- rbcnorm returns a vector of random variables generated from the bcnorm distribution. Depending on what was provided in mu and sigma, this can be either a vector or a matrix or an array.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Box, G. E., & Cox, D. R. (1964). An Analysis of Transformations. Journal of the Royal Statistical Society. Series B (Methodological), 26(2), 211–252. Retrieved from https://www.jstor.org/stable/2984418

Examples

```
x <- dbcnorm(c(-1000:1000)/200, 0, 1, 1)
plot(c(-1000:1000)/200, x, type="l")

x <- pbcnorm(c(-1000:1000)/200, 0, 1, 1)
plot(c(-1000:1000)/200, x, type="l")

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

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

`determination` *Determination coefficients*

Description

Function produces determination coefficient for the provided data

Usage

```
determination(xreg, bruteforce = TRUE, ...)

determ(xreg, bruteforce = TRUE, ...)
```

Arguments

`xreg`	Data frame or a matrix, containing the exogenous variables.
`bruteforce`	If `TRUE`, then all the variables will be used for the regression construction (sink regression). If the number of observations is smaller than the number of series, the function will use stepwise function and select only meaningful variables. So the reported values will be based on stepwise regressions for each variable.
`...`	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).

See details in the vignette "Marketing analytics with greybox": `vignette("maUsingGreybox","greybox")`

Value

Function returns the vector of determination coefficients.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

cor, mcor, 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("x1","x2","x3","Noise")
determination(xreg)
```

---

dfnorm *Folded Normal Distribution*

Description

Density, cumulative distribution, quantile functions and random number generation for the folded normal distribution with the location parameter mu and the scale sigma (which corresponds to standard deviation in normal distribution).

Usage

```
dfnorm(q, mu = 0, sigma = 1, log = FALSE)

pfnorm(q, mu = 0, sigma = 1)

qfnorm(p, mu = 0, sigma = 1)

rfnorm(n = 1, mu = 0, sigma = 1)
```

Arguments

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

Details

The distribution has the following density function:

f(x) = 1/sqrt(2 pi) (exp(-(x-mu)^2 / (2 sigma^2)) + exp(-(x+mu)^2 / (2 sigma^2)))

Both pfnorm and qfnorm are returned for the lower tail of the distribution.

Value

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

- `dfnorm` returns the density function value for the provided parameters.
- `pfnorm` returns the value of the cumulative function for the provided parameters.
- `qfnorm` returns quantiles of the distribution. Depending on what was provided in `p`, `mu` and `sigma`, this can be either a vector or a matrix, or an array.
- `rfnorm` returns a vector of random variables generated from the fnorm distribution. Depending on what was provided in `mu` and `sigma`, this can be either a vector or a matrix or an array.

Author(s)

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

References

- Wikipedia page on folded normal distribution: https://en.wikipedia.org/wiki/Folded_normal_distribution.

Examples

```
x <- dfnorm(c(-1000:1000)/200, 0, 1)
plot(x, type="l")

x <- pfnorm(c(-1000:1000)/200, 0, 1)
plot(x, type="l")

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

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

---

dlaplace *Laplace Distribution*

Description

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

Usage

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

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

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

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

Arguments

q	vector of quantiles.
mu	vector of location parameters (means).
scale	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 scale=1, the Laplace distribution becomes standardized. The distribution has the following density function:

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

Both `plaplace` and `qlaplace` are returned for the lower tail of the distribution.

Value

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

- `dlaplace` returns the density function value for the provided parameters.
- `plaplace` returns the value of the cumulative function for the provided parameters.
- `qlaplace` returns quantiles of the distribution. Depending on what was provided in `p`, `mu` and `scale`, this can be either a vector or a matrix, or an array.
- `rlaplace` returns a vector of random variables generated from the Laplace distribution. Depending on what was provided in `mu` and `scale`, 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(-100:100)/10, 0, 1)
plot(x, type="l")

x <- plaplace(c(-100:100)/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 number generation for the S distribution with the location parameter mu and a scaling parameter scale.

Usage

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

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

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

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

Arguments

q	vector of quantiles.
mu	vector of location parameters (means).
scale	vector of scaling parameter (which are equal 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 scale=1 (this is because scale=ham/2). The distribution has the following density function:

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

The S distribution has fat tails and large excess.

Both ps and qs are returned for the lower tail of the distribution.

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 scale, 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 scale, 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)
```

---

dtplnorm *Three Parameter Log Normal Distribution*

Description

Density, cumulative distribution, quantile functions and random number generation for the 3 parameter log normal distribution with the location parameter mu, scale sigma (which corresponds to standard deviation in normal distribution) and shifting parameter shift.

Usage

```
dtplnorm(q, mu = 0, sigma = 1, shift = 0, log = FALSE)

ptplnorm(q, mu = 0, sigma = 1, shift = 0)

qtplnorm(p, mu = 0, sigma = 1, shift = 0)

rtplnorm(n = 1, mu = 0, sigma = 1, shift = 0)
```

Arguments

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

Details

The distribution has the following density function:

f(x) = 1/(x-a) 1/sqrt(2 pi) exp(-(log(x-a)-mu)^2 / (2 sigma^2))

Both ptplnorm and qtplnorm are returned for the lower tail of the distribution.

The function is based on the lnorm functions from stats package, introducing the shift parameter.

Value

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

- dtplnorm returns the density function value for the provided parameters.
- ptplnorm returns the value of the cumulative function for the provided parameters.
- qtplnorm returns quantiles of the distribution. Depending on what was provided in p, mu and sigma, this can be either a vector or a matrix, or an array.
- rtplnorm returns a vector of random variables generated from the tplnorm distribution. Depending on what was provided in mu and sigma, this can be either a vector or a matrix or an array.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Sangal, B. P., & Biswas, A. K. (1970). The 3-Parameter Distribution Applications in Hydrology. Water Resources Research, 6(2), 505–515. https://doi.org/10.1029/WR006i002p00505

Examples

```
x <- dtplnorm(c(-1000:1000)/200, 0, 1, 1)
plot(c(-1000:1000)/200, x, type="l")

x <- ptplnorm(c(-1000:1000)/200, 0, 1, 1)
plot(c(-1000:1000)/200, x, type="l")

qtplnorm(c(0.025,0.975), 0, c(1,2), 1)
```

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

---

errorType *Functions that extracts type of error from the model*

Description

This function allows extracting error type from any model.

Usage

```
errorType(object, ...)
```

Arguments

object	Model estimated using one of the functions of smooth package.
...	Currently nothing is accepted via ellipsis.

Details

errorType extracts the type of error from the model (either additive or multiplicative).

Value

Either "A" for additive error or "M" for multiplicative. All the other functions return strings of character.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

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 <- alm(y~x1+x2,as.data.frame(xreg))

errorType(ourModel)
```

graphmaker *Linear graph construction function*

Description

The function makes a standard linear graph using the provided actuals and forecasts.

Usage

```
graphmaker(actuals, forecast, fitted = NULL, lower = NULL,
  upper = NULL, level = NULL, legend = TRUE, cumulative = FALSE,
  vline = TRUE, parReset = TRUE, ...)
```

Arguments

actuals	The vector of actual values
forecast	The vector of forecasts. Should be ts object that starts at the end of fitted values.
fitted	The vector of fitted values.
lower	The vector of lower bound values of a prediction interval. Should be ts object that start at the end of fitted values.
upper	The vector of upper bound values of a prediction interval. Should be ts object that start at the end of fitted values.
level	The width of the prediction interval.
legend	If TRUE, the legend is drawn.
cumulative	If TRUE, then the forecast is treated as cumulative and value per period is plotted.
vline	Whether to draw the vertical line, splitting the in-sample and the holdout sample.
parReset	Whether to reset par() after plotting things or not. If FALSE then you can add elements to the plot (e.g. additional lines).
...	Other parameters passed to plot() function.

Details

Function uses the provided data to construct a linear graph. It is strongly advised to use ts objects to define the start of each of the vectors. Otherwise the data may be plotted incorrectly.

Value

Function does not return anything.

Author(s)

Ivan Svetunkov

See Also

ts

Examples

```
x <- rnorm(100,0,1)
values <- forecast(arima(x),h=10,level=0.95)

graphmaker(x,values$mean,fitted(values))
graphmaker(x,values$mean,fitted(values),legend=FALSE)
graphmaker(x,values$mean,fitted(values),values$lower,values$upper,level=0.95)
graphmaker(x,values$mean,fitted(values),values$lower,values$upper,level=0.95,legend=FALSE)

# Produce the necessary ts objects from an arbitrary vectors
actuals <- ts(c(1:10), start=c(2000,1), frequency=4)
forecast <- ts(c(11:15),start=end(actuals)[1]+end(actuals)[2]*deltat(actuals),
               frequency=frequency(actuals))
graphmaker(actuals,forecast)

# This should work as well
graphmaker(c(1:10),c(11:15))

# This way you can add additional elements to the plot
graphmaker(c(1:10),c(11:15), parReset=FALSE)
points(c(1:15))
# But don't forget to do dev.off() in order to reset the plotting area afterwards
```

---

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.
- pointLik - point likelihood of the function.

- pAIC, pAICc, pBIC, pBICc - point versions of respective information criteria.
- determination - Coefficients of determination between different exogenous variables.
- alm - Advanced Linear Model - regression, estimated using likelihood with specified distribution (e.g. Laplace or Chi-Squared).
- 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.
- xregTransformer - Function produces mathematical transformations of the variables, such as taking logarithms, square roots etc.
- xregMultiplier - Function produces cross-products of the matrix of the provided variables.
- lmCombine - Function combines lm models from the estimated based on information criteria weights.
- lmDynamic - Dynamic regression based on point AIC.
- ro - Rolling origin evaluation.
- qlaplace, dlaplace, plaplace, rlaplace - Laplace distribution and the respective functions.
- qfnorm, dfnorm, pfnorm, rfnorm - Folded normal distribution and the respective functions.
- qs, ds, ps, rs - S distribution and the respective functions.
- qtplnorm, dtplnorm, ptplnorm, rtplnorm - Three parameter log normal distribution and the respective functions.
- qbcnorm, dbcnorm, pbcnorm, rbcnorm - Box-Cox normal 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)
```

hm *Half moment of a distribution and its derivatives.*

Description

hm function estimates half moment from some predefined constant C. ham estimates half absolute moment. Finally, cbias function returns bias based on hm.

Usage

```
hm(x, C = mean(x), ...)

ham(x, C = mean(x), ...)

cbias(x, C = mean(x), ...)
```

Arguments

x	A variable based on which HM is estimated.
C	Centering parameter.
...	Other parameters passed to mean function.

Details

NA values of x are excluded on the first step of calculation.

Value

A complex variable is returned for hm function and real values are returned for cbias and ham.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

Examples

```
x <- rnorm(100,0,1)
hm(x)
ham(x)
cbias(x)
```

`is.greybox` *Greybox classes checkers*

Description

Functions to check if an object is of the specified class

Usage

```
is.greybox(x)

is.alm(x)

is.greyboxC(x)

is.greyboxD(x)

is.rollingOrigin(x)

is.rmc(x)
```

Arguments

`x`	The object to check.

Details

The list of functions includes:

- `is.greybox()` tests if the object was produced by a greybox function (e.g. alm / stepwise / lmCombine / lmDynamic);
- `is.alm()` tests if the object was produced by `alm()` function;
- `is.greyboxC()` tests if the object was produced by `lmCombine()` function;
- `is.greyboxD()` tests if the object was produced by `lmDynamic()` function;
- `is.rmc()` tests if the object was produced by `rmc()` function;
- `is.rollingOrigin()` tests if the object was produced by `ro()` function.

Value

`TRUE` if this is the specified class and `FALSE` otherwise.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

Examples

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

ourModel <- alm(y~x1+x2, xreg, distribution="dnorm")

is.alm(ourModel)
is.greybox(ourModel)
is.greyboxC(ourModel)
is.greyboxD(ourModel)
```

---

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, distribution = c("dnorm",
  "dfnorm", "dlnorm", "dlaplace", "ds", "dchisq", "dlogis", "plogis",
  "pnorm"), parallel = FALSE, ...)
```

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.
distribution	Distribution to pass to alm().
parallel	If TRUE, then the model fitting is done in parallel. WARNING! Packages foreach and either doMC (Linux and Mac only) or doParallel are needed in order to run the function in parallel.
...	Other parameters passed to alm().

Details

The algorithm uses alm() to fit different models and then combines the models based on the selected IC. The parameters are combined so that if they are not present in some of models, it is assumed that they are equal to zero. Thus, there is a shrinkage effect in the combination.

Some details and examples of application are also given in the vignette "Greybox": `vignette("greybox","greybox")`

Value

Function returns `model` - the final model of the class "greyboxC". The list of variables:

- coefficients - combined parameters of the model,
- se - combined standard errors of the parameters of the model,
- fitted - the fitted values,
- residuals - residual of the model,
- distribution - distribution used in the estimation,
- logLik - combined log-likelihood of the model,
- IC - the values of the combined information criterion,
- df.residual - number of degrees of freedom of the residuals of the combined model,
- df - number of degrees of freedom of the combined model,
- importance - importance of the parameters,
- combination - the table, indicating which variables were used in every model construction and what were the weights for each model.

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)
predict(ourModel,outSample)
plot(predict(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(predict(ourModel, outSample))

# Combine in parallel - should increase speed in case of big data
## Not run: ourModel <- lmCombine(inSample, ic="BICc", bruteforce=TRUE, parallel=TRUE)
summary(ourModel)
plot(predict(ourModel, outSample))
## End(Not run)
```

`lmDynamic` *Combine regressions based on point information criteria*

Description

Function combines parameters of linear regressions of the first variable on all the other provided data using pAIC weights

Usage

```
lmDynamic(data, ic = c("AICc", "AIC", "BIC", "BICc"),
  bruteforce = FALSE, silent = TRUE, distribution = c("dnorm",
  "dfnorm", "dlnorm", "dlaplace", "ds", "dchisq", "dlogis", "plogis",
  "pnorm"), parallel = FALSE, ...)
```

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.
`distribution`	Distribution to pass to `alm()`.

`parallel` If `TRUE`, then the model fitting is done in parallel. WARNING! Packages `foreach` and either `doMC` (Linux and Mac only) or `doParallel` are needed in order to run the function in parallel.

`...` Other parameters passed to `alm()`.

Details

The algorithm uses alm() to fit different models and then combines the models based on the selected point IC. This is a dynamic counterpart of lmCombine function.

Some details and examples of application are also given in the vignette "Greybox": `vignette("greybox","greybox")`

Value

Function returns `model` - the final model of the class "greyboxD", which includes time varying parameters and dynamic importance of each variable. The list of variables:

- coefficients - mean (over time) parameters of the model,
- se - mean standard errors of the parameters of the model,
- fitted - the fitted values,
- residuals - residual of the model,
- distribution - distribution used in the estimation,
- logLik - mean (over time) log-likelihood of the model,
- IC - dynamic values of the information criterion (pIC),
- df.residual - mean number of degrees of freedom of the residuals of the model,
- df - mean number of degrees of freedom of the model,
- importance - dynamic importance of the parameters,
- call - call used in the function,
- rank - rank of the combined model,
- data - the data used in the model,
- mu - the location value of the distribution,
- scale - the scale parameter if alm() was used,
- coefficientsDynamic - table with parameters of the model, varying over the time,
- df.residualDynamic - dynamic df.residual,
- dfDynamic - dynamic df.

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

stepwise,lmCombine

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 <- lmDynamic(inSample,bruteforce=TRUE)
predict(ourModel,outSample)
plot(predict(ourModel,outSample))
```

MAE *Error measures*

Description

Functions allow to calculate different types of errors for point and interval predictions:

1. MAE - Mean Absolute Error,
2. MSE - Mean Squared Error,
3. MRE - Mean Root Error (Kourentzes, 2014),
4. MIS - Mean Interval Score (Gneiting & Raftery, 2007),
5. MPE - Mean Percentage Error,
6. MAPE - Mean Absolute Percentage Error (See Svetunkov, 2017 for the critique),
7. MASE - Mean Absolute Scaled Error (Hyndman & Koehler, 2006)),
8. rMAE - Relative Mean Absolute Error (Davydenko & Fildes, 2013),
9. rRMSE - Relative Root Mean Squared Error,
10. rAME - Relative Absolute Mean Error,
11. rMIS - Relative Mean Interval Score,
12. sMSE - Scaled Mean Squared Error (Petropoulos & Kourentzes, 2015),
13. sPIS- Scaled Periods-In-Stock (Wallstrom & Segerstedt, 2010),
14. sCE - Scaled Cumulative Error,
15. sMIS - Scaled Mean Interval Score.

Usage

```
MAE(actual, forecast)

MSE(actual, forecast)

MRE(actual, forecast)

MIS(actual, lower, upper, level = 0.95)

MPE(actual, forecast)

MAPE(actual, forecast)

MASE(actual, forecast, scale)

rMAE(actual, forecast, benchmark)

rRMSE(actual, forecast, benchmark)

rAME(actual, forecast, benchmark)

rMIS(actual, lower, upper, benchmarkLower, benchmarkUpper, level = 0.95)

RelMAE(actual, forecast, benchmark)

RelRMSE(actual, forecast, benchmark)

RelAME(actual, forecast, benchmark)

RelMIS(actual, lower, upper, benchmarkLower, benchmarkUpper,
  level = 0.95)

sMSE(actual, forecast, scale)

sPIS(actual, forecast, scale)

sCE(actual, forecast, scale)

sMIS(actual, lower, upper, scale, level = 0.95)
```

Arguments

actual	The vector or matrix of actual values.
forecast	The vector or matrix of forecasts values.
lower	The lower bound of the prediction interval.
upper	The upper bound of the prediction interval.
level	The confidence level of the constructed interval.

scale	The value that should be used in the denominator of MASE. Can be anything but advised values are: mean absolute deviation of in-sample one step ahead Naive error or mean absolute value of the in-sample actuals.
benchmark	The vector or matrix of the forecasts of the benchmark model.
benchmarkLower	The lower bound of the prediction interval of the benchmark model.
benchmarkUpper	The upper bound of the prediction interval of the benchmark model.

Details

In case of sMSE, scale needs to be a squared value. Typical one – squared mean value of in-sample actuals.

If all the measures are needed, then measures function can help.

There are several other measures, see details of pinball and hm.

Value

All the functions return the scalar value.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Kourentzes N. (2014). The Bias Coefficient: a new metric for forecast bias https://kourentzes.com/forecasting/2014/12/17/the-bias-coefficient-a-new-metric-for-forecast-bias/
- Svetunkov, I. (2017). Naughty APEs and the quest for the holy grail. https://forecasting.svetunkov.ru/en/2017/07/29/naughty-apes-and-the-quest-for-the-holy-grail/
- Fildes R. (1992). The evaluation of extrapolative forecasting methods. International Journal of Forecasting, 8, pp.81-98.
- Hyndman R.J., Koehler A.B. (2006). Another look at measures of forecast accuracy. International Journal of Forecasting, 22, pp.679-688.
- Petropoulos F., Kourentzes N. (2015). Forecast combinations for intermittent demand. Journal of the Operational Research Society, 66, pp.914-924.
- Wallstrom P., Segerstedt A. (2010). Evaluation of forecasting error measurements and techniques for intermittent demand. International Journal of Production Economics, 128, pp.625-636.
- Davydenko, A., Fildes, R. (2013). Measuring Forecasting Accuracy: The Case Of Judgmental Adjustments To Sku-Level Demand Forecasts. International Journal of Forecasting, 29(3), 510-522. https://doi.org/10.1016/j.ijforecast.2012.09.002
- Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association, 102(477), 359–378. https://doi.org/10.1198/016214506000001437

See Also

pinball, hm, measures

Examples

```
y <- rnorm(100,10,2)
testForecast <- rep(mean(y[1:90]),10)

MAE(y[91:100],testForecast)
MSE(y[91:100],testForecast)

MPE(y[91:100],testForecast)
MAPE(y[91:100],testForecast)

# Measures from Petropoulos & Kourentzes (2015)
MASE(y[91:100],testForecast,mean(abs(y[1:90])))
sMSE(y[91:100],testForecast,mean(abs(y[1:90]))^2)
sPIS(y[91:100],testForecast,mean(abs(y[1:90])))
sCE(y[91:100],testForecast,mean(abs(y[1:90])))

# Original MASE from Hyndman & Koehler (2006)
MASE(y[91:100],testForecast,mean(abs(diff(y[1:90]))))

testForecast2 <- rep(y[91],10)
# Relative measures, from and inspired by Davydenko & Fildes (2013)
rMAE(y[91:100],testForecast2,testForecast)
rRMSE(y[91:100],testForecast2,testForecast)
rAME(y[91:100],testForecast2,testForecast)

#### Measures for the prediction intervals
# An example with mtcars data
ourModel <- alm(mpg~., mtcars[1:30,], distribution="dnorm")
ourBenchmark <- alm(mpg~1, mtcars[1:30,], distribution="dnorm")

# Produce predictions with the interval
ourForecast <- predict(ourModel, mtcars[-c(1:30),], interval="p")
ourBenchmarkForecast <- predict(ourBenchmark, mtcars[-c(1:30),], interval="p")

MIS(mtcars$mpg[-c(1:30)],ourForecast$lower,ourForecast$upper,0.95)
sMIS(mtcars$mpg[-c(1:30)],ourForecast$lower,ourForecast$upper,mean(mtcars$mpg[1:30]),0.95)
rMIS(mtcars$mpg[-c(1:30)],ourForecast$lower,ourForecast$upper,
       ourBenchmarkForecast$lower,ourBenchmarkForecast$upper,0.95)

### Also, see pinball function for other measures for the intervals
```

---

mcor *Multiple correlation*

Description

Function calculates multiple correlation between y and x, constructing a linear regression model

Usage

```
mcor(x, y, use = c("na.or.complete", "complete.obs", "everything",
  "all.obs"))
```

Arguments

x	Either data.frame or a matrix
y	The numerical variable.
use	What observations to use. See cor function for details. The only option that is not available here is "pairwise.complete.obs".

Details

This is based on the linear regression model with the set of variables in x. The returned value is just a coefficient of multiple correlation from regression, the F-statistics of the model (thus testing the null hypothesis that all the parameters are equal to zero), the associated p-value and the degrees of freedom.

See details in the vignette "Marketing analytics with greybox": `vignette("maUsingGreybox","greybox")`

Value

The following list of values is returned:

- valueThe value of the coefficient;
- statisticThe value of F-statistics associated with the parameter;
- p.valueThe p-value of F-statistics associated with the parameter;
- df.residualThe number of degrees of freedom for the residuals;
- dfThe number of degrees of freedom for the data.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

table, tableplot, spread, cramer, association

Examples

```
mcor(mtcars$am, mtcars$mpg)
```

`measures` *Error measures for the provided forecasts*

Description

Function calculates several error measures using the provided forecasts and the data for the holdout sample.

Usage

```
measures(holdout, forecast, actual, digits = NULL)
```

Arguments

`holdout`	The vector of the holdout values.
`forecast`	The vector of forecasts produced by a model.
`actual`	The vector of actual in-sample values.
`digits`	Number of digits of the output. If `NULL` then no rounding is done.

Value

The functions returns the named vector of errors:

- MAE,
- MSE
- MPE,
- MAPE,
- MASE,
- sMAE,
- sMSE,
- sCE,
- rMAE,
- rRMSE,
- rAME,
- cbias,
- sPIS.

For the details on these errors, see Errors.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Svetunkov, I. (2017). Naughty APEs and the quest for the holy grail. https://forecasting.svetunkov.ru/en/2017/07/29/naughty-apes-and-the-quest-for-the-holy-grail/
- Fildes R. (1992). The evaluation of extrapolative forecasting methods. International Journal of Forecasting, 8, pp.81-98.
- Hyndman R.J., Koehler A.B. (2006). Another look at measures of forecast accuracy. International Journal of Forecasting, 22, pp.679-688.
- Petropoulos F., Kourentzes N. (2015). Forecast combinations for intermittent demand. Journal of the Operational Research Society, 66, pp.914-924.
- Wallstrom P., Segerstedt A. (2010). Evaluation of forecasting error measurements and techniques for intermittent demand. International Journal of Production Economics, 128, pp.625-636.
- Davydenko, A., Fildes, R. (2013). Measuring Forecasting Accuracy: The Case Of Judgmental Adjustments To Sku-Level Demand Forecasts. International Journal of Forecasting, 29(3), 510-522. https://doi.org/10.1016/j.ijforecast.2012.09.002

Examples

```
y <- rnorm(100,10,2)
ourForecast <- rep(mean(y[1:90]),10)

measures(y[91:100],ourForecast,y[1:90],digits=5)
```

`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)
```

---

pAIC *Point AIC*

Description

This function returns a vector of AIC values for the in-sample observations

Usage

```
pAIC(object, ...)

pAICc(object, ...)

pBIC(object, ...)

pBICc(object, ...)
```

Arguments

object	Time series model.
...	Some stuff.

Details

This is based on pointLik function. The formula for this is: pAIC_t = 2 * k - 2 * T * l_t , where k is the number of parameters, T is the number of observations and l_t is the point likelihood. This way we preserve the property that AIC = mean(pAIC).

Value

The function returns the vector of point AIC values.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

pointLik

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 <- alm(y~x1+x2,as.data.frame(xreg))

pAICValues <- pAIC(ourModel)

mean(pAICValues)
AIC(ourModel)
```

---

pinball *Pinball function*

Description

The function returns the value from the pinball function for the specified level and the type of loss

Usage

```
pinball(holdout, forecast, level, loss = 1)
```

Arguments

holdout	The vector or matrix of the holdout values.
forecast	The forecast of prediction interval (should be the same length as the holdout).
level	The level of the prediction interval associated with the forecast.
loss	The type of loss to use. The number which corresponds to L1, L2 etc.

Value

The function returns the scalar value.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

Examples

```
# An example with mtcars data
ourModel <- alm(mpg~., mtcars[1:30,], distribution="dnorm")

# Produce predictions with the interval
ourForecast <- predict(ourModel, mtcars[-c(1:30),], interval="p")

# Pinball with the L1 (quantile value)
pinball(mtcars$mpg[-c(1:30)],ourForecast$upper,level=0.975,loss=1)
pinball(mtcars$mpg[-c(1:30)],ourForecast$lower,level=0.025,loss=1)

# Pinball with the L2 (expectile value)
pinball(mtcars$mpg[-c(1:30)],ourForecast$upper,level=0.975,loss=2)
pinball(mtcars$mpg[-c(1:30)],ourForecast$lower,level=0.025,loss=2)
```

---

pointLik *Point likelihood values*

Description

This function returns a vector of logarithms of likelihoods for each observation

Usage

```
pointLik(object, ...)
```

Arguments

object	Time series model.
...	Some stuff.

Details

Instead of taking the expected log-likelihood for the whole series, this function calculates the individual value for each separate observation. Note that these values are biased, so you would possibly need to take number of degrees of freedom into account in order to have an unbiased estimator.

This value is based on the general likelihood (not its concentrated version), so the sum of these values may slightly differ from the output of logLik.

Value

This function returns a vector.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

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 <- alm(y~x1+x2,as.data.frame(xreg))

pointLik(ourModel)

# Bias correction
pointLik(ourModel) - nparam(ourModel)

# Bias correction in AIC style
2*(nparam(ourModel)/nobs(ourModel) - pointLik(ourModel))

# BIC calculation based on pointLik
log(nobs(ourModel))*nparam(ourModel) - 2*sum(pointLik(ourModel))
```

---

polyprod *This function calculates parameters for the polynomials*

Description

The function accepts two vectors with the parameters for the polynomials and returns the vector of parameters after their multiplication. This can be especially useful, when working with ARIMA models.

Usage

```
polyprod(x, y)
```

Arguments

x	The vector of parameters of the first polynomial.
y	The vector of parameters of the second polynomial.

Value

The function returns a matrix with one column with the parameters for the polynomial, starting from the 0-order.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

convolve

Examples

```
polyprod(c(1,-2,-1),c(1,0.5,0.3))
```

---

predict.alm *Forecasting using greybox functions*

Description

The functions allow producing forecasts based on the provided model and newdata.

Usage

```
## S3 method for class 'alm'
predict(object, newdata = NULL, interval = c("none",
  "confidence", "prediction"), level = 0.95, side = c("both", "upper",
  "lower"), ...)

## S3 method for class 'greybox'
predict(object, newdata = NULL, interval = c("none",
  "confidence", "prediction"), level = 0.95, side = c("both", "upper",
  "lower"), ...)

## S3 method for class 'greybox'
forecast(object, newdata = NULL, h = NULL, ...)

## S3 method for class 'alm'
forecast(object, newdata = NULL, h = NULL, ...)
```

Arguments

object	Time series model for which forecasts are required.
newdata	Forecast horizon
interval	Type of intervals to construct: either "confidence" or "prediction". Can be abbreviated
level	Confidence level. Defines width of prediction interval.
side	What type of interval to produce: "both" - produces both lower and upper bounds of the interval, "upper" - upper only, "lower" - respectively lower only. In the "both" case the probability is split into two parts: ((1-level)/2, (1+level)/2). When "upper" is specified, then the intervals for (0, level) are constructed Finally, with "lower" the interval for (1-level, 1) is returned.
...	Other arguments.
h	The forecast horizon.

Details

predict produces predictions for the provided model and newdata. If newdata is not provided, then the data from the model is extracted and the fitted values are reproduced. This might be useful when confidence / prediction intervals are needed for the in-sample values.

forecast function produces forecasts for h steps ahead. There are four scenarios in this function:

1. If the newdata is not provided, then it will produce forecasts of the explanatory variables to the horizon h (using es from smooth package or using Naive if smooth is not installed) and use them as newdata.
2. If h and newdata are provided, then the number of rows to use will be regulated by h.
3. If h is NULL, then it is set equal to the number of rows in newdata.
4. If both h and newdata are not provided, then it will use the data from the model itself, reproducing the fitted values.

After forming the newdata the forecast function calls for predict, so you can provide parameters interval, level and side in the call for forecast.

Value

predict.greybox() returns object of class "predict.greybox", which contains:

- model - the estimated model.
- mean - the expected values.
- fitted - fitted values of the model.
- lower - lower bound of prediction / confidence intervals.
- upper - upper bound of prediction / confidence intervals.
- level - confidence level.
- newdata - the data provided in the call to the function.
- variances - conditional variance for the holdout sample. In case of interval="prediction" includes variance of the error.

`predict.alm()` is based on `predict.greybox()` and returns object of class "predict.alm", which in addition contains:

- `location` - the location parameter of the distribution.
- `scale` - the scale parameter of the distribution.
- `distribution` - name of the fitted distribution.

`forecast()` functions return the same "predict.alm" and "predict.greybox" classes, with the same set of output variables.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

predict.lm, forecast

Examples

```
xreg <- cbind(rlaplace(100,10,3),rnorm(100,50,5))
xreg <- cbind(100+0.5*xreg[,1]-0.75*xreg[,2]+rlaplace(100,0,3),xreg,rnorm(100,300,10))
colnames(xreg) <- c("y","x1","x2","Noise")
inSample <- xreg[1:80,]
outSample <- xreg[-c(1:80),]

ourModel <- alm(y~x1+x2, inSample, distribution="dlaplace")

predict(ourModel,outSample)
predict(ourModel,outSample,interval="c")

plot(predict(ourModel,outSample,interval="p"))
plot(forecast(ourModel,h=10,interval="p"))
```

---

rmc *Regression for Multiple Comparison*

Description

RMC stands for "Regression for Multiple Comparison", referring to the comparison of forecasting methods. This is a parametric test for the comparison of means of several distributions This test is a parametric counterpart of Nemenyi / MCB test (Demsar, 2006) and uses asymptotic properties of regression models. It relies on distributional assumptions about the provided data. For instance, if the mean forecast errors are used, then it is safe to assume that the regression model constructed on them will have symmetrically distributed residuals, thus normal regression can be used for the parameters estimation.

Usage

```
rmc(data, distribution = c("dlnorm", "dnorm", "dfnorm"), level = 0.95,
  outplot = c("mcb", "lines", "none"), select = NULL, ...)
```

Arguments

data	Matrix or data frame with observations in rows and variables in columns.
distribution	Type of the distribution to use. This value is passed to alm() function. "dlnorm" would lead to the alm with log normal distribution. If this is a clear forecast error, then "dnorm" might be appropriate, leading to a simple Gausian linear regression. "dfnorm" would lead to an alm model with folded normal distribution. You can try some other distributions, but don't expect anything meaningful.
level	The width of the confidence interval. Default is 0.95.
outplot	What type of plot to use after the calculations. This can be either "MCB" ("mcb"), or "Vertical lines" ("lines"), or nothing ("none"). You can also use plot method on the produced object in order to get the same effect.
select	What column of data to highlight on the plot. If NULL, then the method with the lowest value is selected.
...	Other parameters passed to plot function

Details

The test constructs the regression model of the type:

y = b' X + e,

where y is the vector of the provided data (as.vector(data)), X is the matrix of dummy variables for each column of the data (forecasting method), b is the vector of coefficients for the dummies and e is the error term of the model.

Depending on the provided data, it might make sense to use different types of regressions. The default one is the log normal distribution for the relative error measures. The type of distribution is regulated with distribution and is restricted by the values of it from the alm function.

The advisable error measures to use in the test are relative measures, such as RelMAE, RelRMSE, RelAME. They are unbiased and their logarithms are symmetrically distributed (Davydenko & Fildes, 2013). Although their distributions are not log normal, given the typically large samples of datasets, the Central Limit Theorem helps in the adequate construction of the confidence intervals for the parameters.

The test is equivalent to Nemenyi test, when applied to the ranks of the error measures on large samples with distribution="dnorm".

There is also a plot() method that allows producing either "mcb" or "lines" style of plot. This can be regulated via plot(x,outplot="lines").

Value

If outplot!="none", then the function plots the results after all the calculations. In case of distribution="dnorm", the closer to zero the intervals are, the better model performs. When distribution="dlnorm" or distribution="dfnorm", the lower, the better.

Function returns a list of a class "rmc", which contains the following variables:

- meanMean values for each method.
- intervalConfidence intervals for each method.
- vlinesCoordinates used for outplot="l", marking the groups of methods.
- groupsThe table containing the groups. TRUE - methods are in the same group, FALSE - they are not.
- methodsSimilar to group parameter, but with a slightly different presentation.
- p.valuep-value for the test of the significance of the model. This is a log-likelihood ratios chi-squared test, comparing the model with the one with intercept only.
- importanceThe weights of the estimated model in comparison with the model with the constant only. 0 means that the constant is better, 1 means that the estimated model is the best.
- levelSignificance level.
- modellm model produced for the calculation of the intervals.
- outplotStyle of the plot to produce.
- selectThe selected variable to highlight.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Demsar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research, 7, 1-30. http://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf
- Davydenko, A., Fildes, R. (2013). Measuring Forecasting Accuracy: The Case Of Judgmental Adjustments To Sku-Level Demand Forecasts. International Journal of Forecasting, 29(3), 510-522. https://doi.org/10.1016/j.ijforecast.2012.09.002
- Hea-Jung Kim (2006) On the Ratio of Two Folded Normal Distributions, Communications in Statistics Theory and Methods, 35:6, 965-977, https://doi.org/10.1080/03610920600672229

See Also

alm

Examples

```
N <- 50
M <- 4
ourData <- matrix(rnorm(N*M,mean=0,sd=1), N, M)
ourData[,2] <- ourData[,2]+4
ourData[,3] <- ourData[,3]+3
ourData[,4] <- ourData[,4]+2
colnames(ourData) <- c("Method A","Method B","Method C - long name","Method D")
rmc(ourData, distribution="dnorm", level=0.95)
# In case of AE-based measures, distribution="dfnorm" should be selected
rmc(abs(ourData), distribution="dfnorm", level=0.95)
```

```
# APE-based measures should not be used in general...

# If RelMAE or RelMSE is used for measuring data, then it makes sense to use
# distribution="dlnorm" for the RelMAE / RelMSE, as it can be approximated by
# log normal distribution, because according to Davydenko & Fildes (2013) the
# logarithms of these measures have symmetric distribution.
ourTest <- rmc((abs(ourData) / rfnorm(N, 0.3, 1)), distribution="dlnorm", level=0.95)
# The exponents of mean values from this function will correspond to the
# geometric means of RelMAE / RelMSE.
exp(ourTest$mean)
# The same is for the intervals:
exp(ourTest$interval)

# You can also reproduce plots in different styles:
plot(ourTest, outplot="lines")

# Or you can use the default "mcb" style and set additional parameters for the plot():
par(mar=c(2,2,4,0)+0.1)
plot(ourTest, main="Four methods")

# The following example should give similar results to Nemenyi test on
# large samples, which compares medians of the distributions:
rmc(t(apply(ourData,1,rank)), distribution="dnorm", level=0.95)

# You can also give a try to SE-based measures with distribution="dchisq":
rmc(ourData^2, distribution="dchisq", level=0.95)
```

`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 model fitting is done in parallel. WARNING! Packages foreach and either doMC (Linux and Mac only) or doParallel are needed in order to run the function in parallel.

... — This is temporary and is needed in order to capture "silent" parameter if it is provided.

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.

For more details and more examples of usage, please see vignette for the function. In order to do that, just run the command: vignette("ro","greybox")

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)
```

---

spread *Construct scatterplot / boxplots for the data*

Description

Function constructs the plots depending on the types of variables in the provided matrix / data frame.

Usage

```
spread(data, histograms = FALSE, log = FALSE, ...)
```

Arguments

data	Either matrix or data frame with the data.
histograms	If TRUE, then the histograms and barplots are produced on the diagonal of the matrix. Otherwise the names of the variables are written there.
log	If TRUE, then the logarithms of all numerical variables are taken.
...	Other parameters passed to the plot function. Currently only "main" parameter is accepted.

Details

If both variables are in metric scale, then the classical scatterplot is constructed. If one of them is either integer (up to 10 values) or categorical (aka 'factor'), then boxplots (with grey dots corresponding to mean values) are constructed. Finally, for the two categorical variables the tableplot is returned (see tableplot function for the details). All of this is packed in a matrix.

See details in the vignette "Marketing analytics with greybox": `vignette("maUsingGreybox","greybox")`

Value

Function does not return anything. It just plots things.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

plot, table, tableplot

Examples

```
### Simple example
spread(mtcars)
spread(mtcars,log=TRUE)
```

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"),
  distribution = c("dnorm", "dfnorm", "dlnorm", "dlaplace", "ds",
  "dchisq", "dlogis", "plogis", "pnorm"), occurrence = c("none",
  "plogis", "pnorm"), ...)
```

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.
distribution	Distribution to pass to alm().
occurrence	what distribution to use for occurrence part. See alm for details.
...	This is temporary and is needed in order to capture "silent" parameter if it is provided.

Details

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

Some details and examples of application are also given in the vignette "Greybox": vignette("greybox","greybox")

Value

Function returns model - the final model of the class "alm". See alm for details of the output.

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,lmCombine

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)

### Mixture distribution of Log Normal and Cumulative Logit
xreg[,1] <- xreg[,1] * round(exp(xreg[,1]-70) / (1 + exp(xreg[,1]-70)),0)
colnames(xreg) <- c("y","x1","x2","Noise")
ourModel <- stepwise(xreg, distribution="dlnorm",
                     occurrence=stepwise(xreg, distribution="plogis"))
summary(ourModel)

### 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))
```

---

tableplot *Construct a plot for categorical variable*

Description

Function constructs a plot for two categorical variables based on table function

Usage

```
tableplot(x, y = NULL, labels = TRUE, legend = TRUE, ...)
```

Arguments

x	First categorical variable. Can be either vector, factor, matrix or a data frame. If y is NULL and x is either matrix of a data frame, then the first two variables of the data will be plotted against each other.
y	Second categorical variable. If not provided, then only x will be plotted.
labels	Whether to print table labels inside the plot or not.
legend	If TRUE, then the legend for the tableplot is drawn.
...	Other parameters passed to the plot function.

Details

The function produces the plot of the table() function with colour densities corresponding to the respective frequencies of appearance. If the value appears more often than the other (e.g. 0.5 vs 0.15), then it will be darker. The frequency of 0 corresponds to the white colour, the frequency of 1 corresponds to the black.

See details in the vignette "Marketing analytics with greybox": vignette("maUsingGreybox","greybox")

Value

Function does not return anything. It just plots things.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

plot,table,spread

Examples

```
tableplot(mtcars$am, mtcars$gear)
```

---

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, gaps = c("auto", "NAs", "zero", "naive",
  "extrapolate"), ...)
```

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.
gaps	Defines how to fill in the gaps in the data. "NAs" will leave missing values, "zero" will substitute them by zeroes, "naive" will use the last / the first actual value, while "extrapolate" will use es function from smooth package (if present, otherwise - naive) in order to fill in values. Finally, "auto" will let the function select between "extrapolate" and "NAs" depending on the length of series.
...	This is temporary and is needed in order to capture "silent" parameter if it is provided.

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, gum 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)
```

xregMultiplier *Exogenous variables cross-products*

Description

Function generates the cross-products of the provided exogenous variables.

Usage

```
xregMultiplier(xreg, silent = TRUE, ...)
```

Arguments

xreg	matrix or data.frame, containing variables that need to be expanded. This matrix needs to contain at least two columns.
silent	If silent=FALSE, then the progress is printed out. Otherwise the function won't print anything in the console.
...	This is temporary and is needed in order to capture "silent" parameter if it is provided.

Details

This function might be useful if you have several variables and want to introduce their cross-products. This might be useful when introducing the interactions between dummy and continuous variables.

Value

ts matrix with the transformed and the original variables is returned.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

es,stepwise,xregExpander,xregTransformer

Examples

```
# Create matrix of two variables and expand it
x <- cbind(rnorm(100,100,1),rnorm(100,50,3))
xregMultiplier(x)
```

`xregTransformer` *Exogenous variables transformer*

Description

Function transforms each variable in the provided matrix or vector, producing non-linear values, depending on the selected pool of functions.

Usage

```
xregTransformer(xreg, functions = c("log", "exp", "inv", "sqrt",
  "square"), 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.
`functions`	Vector of names for functions used.
`silent`	If `silent=FALSE`, then the progress is printed out. Otherwise the function won't print anything in the console.
`...`	This is temporary and is needed in order to capture "silent" parameter if it is provided.

Details

This function could be useful when you want to automatically select the necessary transformations of the variables. This can be used together with `xregDo="select"` in es, ces, gum and ssarima. However, this might be dangerous, as it might lead to the overfitting the data. So be reasonable when you produce the transformed variables.

Value

`ts` matrix with the transformed and the original variables is returned.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

es, stepwise, xregExpander

Examples

```
# Create matrix of two variables and expand it
x <- cbind(rnorm(100,100,1),rnorm(100,50,3))
xregTransformer(x)
```

Index