Package ‘fRegression’

October 13, 2022

Title Rmetrics - Regression Based Decision and Prediction
Date 2022-08-11
Version 4021.83
Description A collection of functions for linear and non-linear regression modelling. It implements a wrapper for several regression models available in the base and contributed packages of R.
Depends R (>= 2.15.1)
Imports fBasics, lmtest, MASS, methods, mgcv, nnet, polspline, stats, timeDate, timeSeries, utils
Suggests RUnit
License GPL (>= 2)
URL https://www.rmetrics.org
BugReports https://r-forge.r-project.org/projects/rmetrics
NeedsCompilation no
Author Diethelm Wuertz [aut], Tobias Setz [aut], Yohan Chalabi [aut], Paul J. Northrop [cre, ctb]
Maintainer Paul J. Northrop <p.northrop@ucl.ac.uk>
Repository CRAN
Date/Publication 2022-08-11 09:40:19 UTC

R topics documented:

fRegression-package ......................................................... 2
coeF-methods ............................................................... 4
fitted-methods ............................................................. 5
formula-methods ............................................................ 6
fREG-class ................................................................. 7
plot-methods ............................................................... 8
Description

The Rmetrics "fRegression" package is a collection of functions for linear and non-linear regression modelling.

Details

Package: fRegression
Type: Package
Version: R 3.0.1
Date: 2014
License: GPL Version 2 or later
Copyright: (c) 1999-2014 Rmetrics Association
Repository: R-FORGE
URL: https://www.rmetrics.org

1 Introduction

Regression modelling, especially linear modelling, LM, is a widely used application in financial engineering. In finance it mostly appears in form that a variable is modelled as a linear or more complex relationship as a function of other variables. For example the decision of buying or selling in a trading model may be triggered by the outcome of a regression model, e.g. neural networks are a well known tool in this field.

2 Fitting Regression Models

Rmetrics has build a unique interface to several regression models available in the base and contributed packages of R. The following regression models are interfaced and available through a common function regFit. The argument use allows to select the desired model:
regFit fits regression models
- lm fits a linear model [stats]
- rlm fits a LM by robust regression [MASS]
- glm fits a generalized linear model [stats]
- gam fits a generalized additive model [mgcv]
- ppr fits a projection pursuit regression model [stats]
- nnet fits a single hidden-layer neural network model [nnet]
- polymars fits an adaptive polynomial spline regression [polspline]

An advantage of the regFit function is, that all the underlying functions of its family can be called with the same list of arguments, and the value returned is always an unique object, an object of class "fREG" with the following slots: @call, @formula, @method, @data, @fit, @residuals, @fitted, @title, and @description.

Furthermore, independent of the selected regression model applied we can use the same S4 methods for all types of regressions. This includes, print, plot, summary, predict, fitted, residuals, coef, vcov, and formula methods.

It is possible to add further regression models to this framework either his own implementations or implementations available through other contributed R packages. Suggestions include biglm, earth amongst others.

2 Simulation of Regression Models

contains a function to simulate artificial regression models, mostly used for testing.

regSim simulates artificial regression model data sets

3 Extractor Functions

These generic functions are:

fitted extracts fitted values from a fitted 'fREG' object
residuals extracts residuals from a fitted 'fREG' object
coef extracts coefficients from a fitted 'fREG' object
formula extracts formula expression from a fitted 'fREG' object
vcov extracts variance-covariance matrix of fitted parameters

4 Forecasting

The function predict returns predicted values based on the fitted model object.

predict forecasts from an object of class 'fREG'
4 Reporting Functions

For printing and plotting use the functions:

```
print         prints the results from a regression fit
plot          plots the results from a regression fit
summary       returns a summary report
```

About Rmetrics:

The fRegression Rmetrics package is written for educational support in teaching "Computational Finance and Financial Engineering" and licensed under the GPL.

---

**Description**

Extracts coefficients from a fitted regression model.

**Methods**

- `object = "ANY"` Generic function.
- `object = "fREG"` Extractor function for coefficients.

**Note**

`coef` is a generic function which extracts the coefficients from objects returned by modeling functions, here the `regFit` and `gregFit` parameter estimation functions.

**Author(s)**

Diethelm Wuertz for the Rmetrics R-port.

**Examples**

```r
## regSim -
x = regSim(model = "LM3", n = 50)

## regFit -
fit = regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")

## coef -
coef(fit)
```
Description
Extracts fitted values from a fitted regression model.

Methods

- `object = "ANY"` Generic function
- `object = "fREG"` Extractor function for fitted values.

Note

fitted is a generic function which extracts fitted values from objects returned by modeling functions, here the `regFit` and `gregFit` parameter estimation functions.

The class of the fitted values is the same as the class of the data input to the function `regFit` or `gregFit`. In contrast the slot `fitted` returns a numeric vector.

Author(s)
Diethelm Wuertz for the Rmetrics R-port.

Examples

```r
## regSim -
x.df = regSim(model = "LM3", n = 50)

## regFit -
# Use data.frame input:
fit = regFit(Y ~ X1 + X2 + X3, data = x.df, use = "lm")

## fitted -
val = slot(fit, "fitted")
head(val)
class(val)
val = fitted(fit)
head(val)
class(val)

## regFit -
# Convert to dummy timeSeries Object:
library(timeSeries)
x.tS = as.timeSeries(x.df)
fit = regFit(Y ~ X1 + X2 + X3, data = x.tS, use = "lm")

## fitted -
val = slot(fit, "fitted")
head(val)
```
class(val)
val = fitted(fit)
head(val)
class(val)

### formula-methods

*Extract Regression Model formula*

#### Description

Extracts formula from a fitted regression model.

#### Methods

- `object = "ANY"`  Generic function
- `object = "fGARCH"`  Formula

#### Note

`formula` is a generic function which extracts the formula expression from objects returned by modeling functions, here the `regFit` and `gregFit` parameter estimation function.

#### Author(s)

Diethelm Wuertz for the Rmetrics R-port.

#### Examples

```r
## regSim -
  x = regSim(model = "LM3", n = 50)

## regFit -
  fit = regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")

## formula -
  formula(fit)
```
The class ‘fREG’ represents a fitted model of an heteroskedastic time series process.

Objects from the Class

Objects can be created by calls of the function regFit. The returned object represents parameter estimates of linear and generalized linear models.

Slots

call: Object of class "call": the call of the garch function.
formula: Object of class "formula": the formula used in parameter estimation.
family: Object of class "character": the family objects provide a convenient way to specify the details of the models used by function gregFit For details we refer to the documentation for the function glm in R's base package on how such model fitting takes place.
method: Object of class "character": a string denoting the regression model in use, i.e. one of those listed in the use argument of the function regFit or gregFit.
data: Object of class "list": a list with at least two entries named x containing the data frame used for the estimation, and data with the object of the rectangular input data.
fit: Object of class "list": a list with the results from the parameter estimation. The entries of the list depend on the selected algorithm, see below.
residuals: Object of class "numeric": a numeric vector with the residual values.
fitted: Object of class "numeric": a numeric vector with the fitted values.
title: Object of class "character": a title string.
description: Object of class "character": a string with a brief description.

Methods

show signature(object = "fREG"): prints an object of class 'fREG'.
plot signature(x = "fREG", y = "missing"): plots an object of class 'fREG'.
summary signature(object = "fREG"): summarizes results and diagnostic analysis of an object of class 'fREG'.
predict signature(object = "fREG"): forecasts mean and volatility from an object of class 'fREG'.
fitted signature(object = "fREG"): extracts fitted values from an object of class 'fREG'.
residuals signature(object = "fREG"): extracts residuals from an object of class 'fREG'.
coef signature(object = "fREG"): extracts fitted coefficients from an object of class 'fREG'.
formula signature(x = "fREG"): extracts formula expression from an object of class 'fREG'.

Author(s)

Diethelm Wuertz and Rmetrics Core Team.
Regression Model Plot Methods

Description

Plots results obtained from a fitted regression model.

Usage

```r
## S4 method for signature 'fREG,missing'
plot(x, which = "ask", ...)  
```

Arguments

- `x`: an object of class 'fREG'.
- `which`: a character string selecting which plot should be displayed. By default `which="ask"` which allows to generate plots interactively.
- `...`: additional arguments to be passed to the underlying plot functions.

Details

The plots are a set of graphs which are common to the regression models implemented in the function `regFit`. This includes linear regression models `use="lm"`, robust linear regression models `use="rlm"`, generalized linear regression models `use = "glm"`, generalized additive regression models `use = "gam"`, projection pursuit regression models `use = "ppr"`, neural network regression models `use = "nnet"`, and polychotomous MARS models `use = "polymars"`. In addition one can also use the original plot functions of the original models, e.g. `plot(slot(object, "fit"))`.

Methods

- `x = "ANY", y = "ANY"` Generic function.
- `x = "fREG", y = "missing"` Plot function to display results obtained from a fitted regression model.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```r
## regSim -
x = regSim(model = "LM3", n = 50)

## regFit -
fit = regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")

## plot -
```
**predict-methods**

**Regression Models Prediction Function**

**Description**

Predicts a time series from a fitted regression model.

**Usage**

```r
## S4 method for signature 'fREG'
predict(object, newdata, se.fit = FALSE, type = "response", ...)
```

**Arguments**

- `newdata` new data.
- `object` an object of class `fREG` as returned from the function `regFit()`.
- `se.fit` a logical flag. Should standard errors be included? By default `FALSE`.
- `type` a character string by default "response".
- `...` arguments to be passed.

**Value**

returns ...

**Methods**

- `object = "ANY"` Generic function
- `object = "fREG"` Predict method for regression models.

**Author(s)**

Diethelm Wuertz for the Rmetrics R-port.

**Examples**

```r
## regSim -
x <- regSim(model = "LM3", n = 50)

## regFit -
fit <- regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")
```
**regFit**

*Regression Modelling*

**Description**

Estimates the parameters of a regression model.

**Usage**

```r
regFit(formula, data, family = gaussian,
       use = c("lm", "rlm", "glm", "gam", "ppr", "nnet", "polymars"),
       title = NULL, description = NULL, ...)
```

**Arguments**

- **data**
  - data is the data frame containing the variables in the model. By default the variables are taken from `environment(formula)`, typically the environment from which `lm` is called.
- **description**
  - a brief description of the project of type character.
- **family**
  - a description of the error distribution and link function to be used in `glm` and `gam` models. See `glm` and `family` for more details.
- **formula**
  - a symbolic description of the model to be fit.
  - A typical `glm` predictor has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a (linear) predictor for `response`. For binomial models the response can also be specified as a factor.
  - A `gam` formula, see also `gam.models`, allows that smooth terms can be added to the right hand side of the formula. See `gam.side.conditions` for details and examples.
- **use**
  - denotes the regression method by a character string used to fit the model. `method` must be one of the strings in the default argument.
  - "lm", for linear regression models,
  - "rlm", for robust linear regression models,
  - "glm" for generalized linear modelling,
  - "gam" for generalized additive modelling,
  - "ppr" for projection pursuit regression,
  - "nnet" for feedforward neural network modelling, and
  - "polymars" for polychotomous MARS.
- **title**
  - a character string which allows for a project title.
- **...**
  - additional optional arguments to be passed to the underlying functions. For details we refer to inspect the following help pages: `lm`, `glm`, `gam`, `ppr`, `polymars`, or `nnet`.
Details

The function `regFit` was created to provide a selection of regression models working together with Rmetrics’ "timeSeries" objects and providing a common S4 object as the returned value. These models include linear modeling, robust linear modeling, generalized linear modeling, generalized additive modelling, projection pursuit regression, neural networks, and polychotomous MARS models.

LM – Linear Modelling:

Univariate linear regression analysis is a statistical methodology that assumes a linear relationship between some predictor variables and a response variable. The goal is to estimate the coefficients and to predict new data from the estimated linear relationship.

R’s base function

```r
lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)
```

is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance, although `aov` may provide a more convenient interface for these.

Rmetrics’ function

```r
regFit(formula, data, use = "lm", ...)
```

calls R’s base function `lm` but with the difference that the data argument, may be any rectangular object which can be transferred by the function `as.data.frame` into a data frame with named columns, e.g. an object of class "timeSeries". The function `regFit` returns an S4 object of class "fREG" whose slot @fit is the object as returned by the function "lm". In addition we have S4 methods `fitted` and `residuals` which allow to retrieve the fitted values and the residuals as objects of same class as defined by the argument data.

The function `plot.lm` provides four plots: a plot of residuals against fitted values, a Scale-Location plot of sqrt(| residuals |) against fitted values, a normal QQ plot, and a plot of Cook’s distances versus row labels.

[stats: lm]

LM – Robust Linear Modelling:

To fit a linear model by robust regression using an M estimator R offers the function

```r
rlm(formula, data, weights, ..., subset, na.action, method = c("M", "MM", "model.frame"), wt.method = c("inv.var", "case"), model = TRUE, x.ret = TRUE, y.ret = FALSE, contrasts = NULL)
```

from package MASS. Again we can use the Rmetrics’ wrapper
regFit(formula, data, use = "rlm", ...)

which allows us to use for example S4 timeSeries objects as input and to get the output as an S4 object with the known slots.

[MASS::rlm]

**GLM – Generalized Linear Models:**

Generalized linear modelling extends the linear model in two directions. (i) with a monotonic differentiable link function describing how the expected values are related to the linear predictor, and (ii) with response variables having a probability distribution from an exponential family.

R's base function from package stats comes with the function

```r
glm(formula, family = gaussian, data, weights, subset, 
na.action, start = NULL, etastart, mustart, offset, 
control = glm.control(...), model = TRUE, method = "glm.fit", 
x = FALSE, y = TRUE, contrasts = NULL, ...)
```

Again we can use the Rmetrics' wrapper

```r
regFit(formula, data, use = "gam", ...)
```

[stats::glm]

**GAM – Generalized Additive Models:**

An additive model generalizes a linear model by smoothing individually each predictor term. A generalized additive model extends the additive model in the same spirit as the generalized linear model extends the linear model, namely for allowing a link function and for allowing non-normal distributions from the exponential family.

[mgcv::gam]

**PPR – Projection Pursuit Regression:**

The basic method is given by Friedman (1984), and is essentially the same code used by S-PLUS's pprreg. It is observed that this code is extremely sensitive to the compiler used. The algorithm first adds up to max.terms, by default ppr.n.terms, ridge terms one at a time; it will use less if it is unable to find a term to add that makes sufficient difference. The levels of optimization, argument optlevel, by default 2, differ in how thoroughly the models are refitted during this process. At level 0 the existing ridge terms are not refitted. At level 1 the projection directions are not refitted, but the ridge functions and the regression coefficients are. Levels 2 and 3 refit all the terms; level 3 is more careful to re-balance the contributions from each regressor at each step and so is a little less likely to converge to a saddle point of the sum of squares criterion. The plot method plots Ridge functions for the projection pursuit regression fit.

[stats::ppr]
POLYMARS – Polychotomous MARS:

The algorithm employed by polymars is different from the MARS(tm) algorithm of Friedman (1991), though it has many similarities. Also the name polymars has been used for this algorithm well before MARS was trademarked.

[polclass:polymars]

NNET – Feedforward Neural Network Regression:

If the response in formula is a factor, an appropriate classification network is constructed; this has one output and entropy fit if the number of levels is two, and a number of outputs equal to the number of classes and a softmax output stage for more levels. If the response is not a factor, it is passed on unchanged to nnet.default. A quasi-Newton optimizer is used, written in C.

[nnet:nnet]

Value

returns an S4 object of class "fREG".

Author(s)

The R core team for the lm functions from R’s base package,
B.R. Ripley for the glm functions from R’s base package,
S.N. Wood for the gam functions from R’s mgcv package,
N.N. for the ppr functions from R’s modreg package,
M. O’ Connors for the polymars functions from R’s ? package,
The R core team for the nnet functions from R’s nnet package,
Diethelm Wuertz for the Rmetrics R-port.

References

Dobson, A.J. (1990); An Introduction to Generalized Linear Models; Chapman and Hall, London.
Green, Silverman (1994); Nonparametric Regression and Generalized Linear Models; Chapman and Hall.


Myers R.H. (1986); *Classical and Modern Regression with Applications*; Duxbury, Boston.


Stone C.J., Hansen M., Kooperberg Ch., and Truong Y.K. (1997); *The use of polynomial splines and their tensor products in extended linear modeling (with discussion).*


Wahba (1990); *Spline Models of Observational Data*; SIAM.


Wood (2000); *Modelling and Smoothing Parameter Estimation with Multiple Quadratic Penalties*; JRSSB 62, 413-428.


Wood (2001); *Thin Plate Regression Splines*.

There exists a vast literature on regression. The references listed above are just a small sample of what is available. The book by Myers’ is an introductory text book that covers discussions of much of the recent advances in regression technology. Seber’s book is at a higher mathematical level and covers much of the classical theory of least squares.

Examples

```r
## regSim -
x <- regSim(model = "LM3", n = 100)

# LM
regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")
# RLM
regFit(Y ~ X1 + X2 + X3, data = x, use = "rlm")
# AM
regFit(Y ~ X1 + X2 + X3, data = x, use = "gam")
# PPR
regFit(Y ~ X1 + X2 + X3, data = x, use = "ppr")
# NNET
regFit(Y ~ X1 + X2 + X3, data = x, use = "nnet")
# POLYMARS
regFit(Y ~ X1 + X2 + X3, data = x, use = "polymars")
```

RegressionTestsInterface

*Regression Tests*
Description

A collection and description of functions to test linear regression models, including tests for higher serial correlations, for heteroskedasticity, for autocorrelations of disturbances, for linearity, and functional relations.

The methods are:

"bg"     Breusch–Godfrey test for higher order serial correlation,
"bp"     Breusch–Pagan test for heteroskedasticity,
"dw"     Durbin–Watson test for autocorrelation of disturbances,
"gq"     Goldfeld–Quandt test for heteroskedasticity,
"harv"   Harvey–Collier test for linearity,
"hmc"    Harrison–McCabe test for heteroskedasticity,
"rain"   Rainbow test for linearity, and
"reset"  Ramsey's RESET test for functional relation.

There is nothing new, it's just a wrapper to the underlying test functions from R's contributed package lmtest. The functions are available as "Builtin" functions. Nevertheless, the user can still install and use the original functions from R's lmtest package.

Usage

lmTest(formula, method = c("bg", "bp", "dw", "gq", "harv", "hmc",
                                   "rain", "reset"), data = list(), ...)

bgTest(formula, order = 1, type = c("Chisq", "F"), data = list())
bpTest(formula, varformula = NULL, studentize = TRUE, data = list())
dwTest(formula, alternative = c("greater", "two.sided", "less"),
        iterations = 15, exact = NULL, tol = 1e-10, data = list())
gqTest(formula, point=0.5, order.by = NULL, data = list())
harvTest(formula, order.by = NULL, data = list())
hmcTest(formula, point = 0.5, order.by = NULL, simulate.p = TRUE,
        nsim = 1000, plot = FALSE, data = list())
rainTest(formula, fraction = 0.5, order.by = NULL, center = NULL,
          data = list())
resetTest(formula, power = 2:3, type = c("fitted", "regressor", "princomp"),
         data = list())

Arguments

- alternative [dwTest] - a character string specifying the alternative hypothesis, either "greater", "two.sided", or "less".
- center [rainTest] - a numeric value. If center is smaller than 1 it is interpreted as percentages of data, i.e. the subset is chosen that n*fraction observations are around observation number n*center. If center is greater than 1 it is interpreted to be the
RegressionTestsInterface

index of the center of the subset. By default center is 0.5. If the Mahalanobis
distance is chosen center is taken to be the mean regressor, but can be specified
to be a k-dimensional vector if k is the number of regressors and should be in
the range of the respective regressors.

data
an optional data frame containing the variables in the model. By default the
variables are taken from the environment which lmTest and the other tests are
called from.

exact
 [dwTest] -
a logical flag. If set to FALSE a normal approximation will be used to compute
the p value, if TRUE the "pan" algorithm is used. The default is to use "pan" if
the sample size is < 100.

formula
a symbolic description for the linear model to be tested.

fraction
 [rainTest] -
a numeric value, by default 0.5. The percentage of observations in the subset is
determined by fraction*n if n is the number of observations in the model.

iterations
 [dwTest] -
an integer specifying the number of iterations when calculating the p-value with
the "pan" algorithm. By default 15.

method
the test method which should be applied.

nsim
 [hmcTest] -
an integer value. Determines how many runs are used to simulate the p value,
by default 1000.

order
 [bgTest] -
an integer. The maximal order of serial correlation to be tested. By default 1.

order.by
 [ggTest][harvTest] -
a formula. A formula with a single explanatory variable like ~ x. Then the
observations in the model are ordered by the size of x. If set to NULL, the default,
the observations are assumed to be ordered (e.g. a time series).
 [rainTest] -
either a formula or a string. A formula with a single explanatory variable like
~ x. The observations in the model are ordered by the size of x. If set to NULL,
the default, the observations are assumed to be ordered (e.g. a time series). If
set to "mahalanobis" then the observations are ordered by their Mahalanobis
distance of the data.

plot
 [hmcTest] -
a logical flag. If TRUE the test statistic for all possible breakpoints is plotted, the
default is FALSE.

point
 [ggTest][hmcTest] -
a numeric value. If point is smaller than 1 it is interpreted as percentages of data,
i.e. n*point is taken to be the (potential) breakpoint in the variances, if n is the
number of observations in the model. If point is greater than 1 it is interpreted
to be the index of the breakpoint. By default 0.5.

power
 [resetTest] -
integers, by default 2:3. A vector of positive integers indicating the powers of
the variables that should be included. By default it is tested for a quadratic or
cubic influence of the fitted response.
simulate.p  [hmcTest] -
a logical. If TRUE, the default, a p-value will be assessed by simulation, otherwise the p-value is NA.

studentize  [bpTest] -
a logical value. If set to TRUE Koenker's studentized version of the test statistic will be used. By default set to TRUE.

tol  [dwTest] -
the tolerance value. Eigenvalues computed have to be greater than tol=1e-10 to be treated as non-zero.

type  [bgTest] -
the type of test statistic to be returned. Either "Chisq" for the Chi-squared test statistic or "F" for the F test statistic.

resetTest  -
a string indicating whether powers of the "fitted" response, the "regressor" variables (factors are left out) or the first principal component, "princomp", of the regressor matrix should be included in the extended model.

varformula  [bpTest] -
a formula describing only the potential explanatory variables for the variance, no dependent variable needed. By default the same explanatory variables are taken as in the main regression model.

...  [regTest] -
additional arguments passed to the underlying lm test. Some of the tests can specify additional optional arguments like for alternative hypothesis, the type of test statistic to be returned, or others. All the optional arguments have default settings.

Details

bg – Breusch Godfrey Test:

Under $H_0$ the test statistic is asymptotically Chi-squared with degrees of freedom as given in parameter. If type is set to "F" the function returns the exact F statistic which, under $H_0$, follows an F distribution with degrees of freedom as given in parameter. The starting values for the lagged residuals in the supplementary regression are chosen to be 0.

bp – Breusch Pagan Test:

The Breusch-Pagan test fits a linear regression model to the residuals of a linear regression model (by default the same explanatory variables are taken as in the main regression model) and rejects if too much of the variance is explained by the additional explanatory variables. Under $H_0$ the test statistic of the Breusch-Pagan test follows a chi-squared distribution with parameter (the number of regressors without the constant in the model) degrees of freedom.

dw – Durbin Watson Test:
The Durbin–Watson test has the null hypothesis that the autocorrelation of the disturbances is 0; it can be tested against the alternative that it is greater than, not equal to, or less than 0 respectively. This can be specified by the alternative argument. The null distribution of the Durbin-Watson test statistic is a linear combination of chi-squared distributions. The p value is computed using a Fortran version of the Applied Statistics Algorithm AS 153 by Farebrother (1980, 1984). This algorithm is called "pan" or "gradsol". For large sample sizes the algorithm might fail to compute the p value; in that case a warning is printed and an approximate p value will be given; this p value is computed using a normal approximation with mean and variance of the Durbin-Watson test statistic.

\[ \text{lmtest:dwtest} \]

**gq – Goldfeld Quandt Test:**

The Goldfeld–Quandt test compares the variances of two submodels divided by a specified breakpoint and rejects if the variances differ. Under $H_0$ the test statistic of the Goldfeld-Quandt test follows an $F$ distribution with the degrees of freedom as given in parameter.

\[ \text{lmtest:gqtest} \]

**harv - Harvey Collier Test:**

The Harvey-Collier test performs a t-test (with parameter degrees of freedom) on the recursive residuals. If the true relationship is not linear but convex or concave the mean of the recursive residuals should differ from 0 significantly.

\[ \text{lmtest:harvtest} \]

**hmc – Harrison McCabe Test:**

The Harrison–McCabe test statistic is the fraction of the residual sum of squares that relates to the fraction of the data before the breakpoint. Under $H_0$ the test statistic should be close to the size of this fraction, e.g. in the default case close to 0.5. The null hypothesis is reject if the statistic is too small.

\[ \text{lmtest:hmctest} \]

**rain – Rainbow Test:**

The basic idea of the Rainbow test is that even if the true relationship is non-linear, a good linear fit can be achieved on a subsample in the "middle" of the data. The null hypothesis is rejected whenever the overall fit is significantly inferior to the fit of the subsample. The test statistic under $H_0$ follows an $F$ distribution with parameter degrees of freedom.

\[ \text{lmtest:raintest} \]

**reset – Ramsey’s RESET Test**

RESET test is popular means of diagnostic for correctness of functional form. The basic assumption is that under the alternative, the model can be written by the regression $y = X\beta + Z\gamma + u$. $Z$ is generated by taking powers either of the fitted response, the regressor variables or the first principal component of $X$. A standard F-Test is then applied to determine whether these additional variables
have significant influence. The test statistic under $H_0$ follows an F distribution with parameter degrees of freedom.

**Value**

A list with class "htest" containing the following components:

- **statistic** the value of the test statistic.
- **parameter** the lag order.
- **p.value** the p-value of the test.
- **method** a character string indicating what type of test was performed.
- **data.name** a character string giving the name of the data.
- **alternative** a character string describing the alternative hypothesis.

**Note**

The underlying `lmtest` package comes with a lot of helpful examples. We highly recommend to install the `lmtest` package and to study the examples given therein.

**Author(s)**

Achim Zeileis and Torsten Hothorn for the `lmtest` package,

Diethelm Wuertz for the Rmetrics R-port.

**References**


Examples

```r
## bg | dw -
# Generate a Stationary and an AR(1) Series:
x = rep(c(1, -1), 50)
y1 = 1 + x + rnorm(100)
# Perform Breusch-Godfrey Test for 1st order serial correlation:
lmTest(y1 ~ x, "bg")
# ... or for fourth order serial correlation:
lmTest(y1 ~ x, "bg", order = 4)
# Compare with Durbin-Watson Test Results:
lmTest(y1 ~ x, "dw")
y2 = filter(y1, 0.5, method = "recursive")
```
```
## bp -
# Generate a Regressor:
x = rep(c(-1, 1), 50)
# Generate heteroskedastic and homoskedastic Disturbances
err1 = rnorm(100, sd = rep(c(1, 2), 50))
err2 = rnorm(100)
# Generate a Linear Relationship:
y1 = 1 + x + err1
y2 = 1 + x + err2
# Perform Breusch-Pagan Test
bp = lmTest(y1 ~ x, "bp")
bp
# Calculate Critical Value for 0.05 Level
qchisq(0.95, bp$parameter)
```
```
## dw -
# Generate two AR(1) Error Terms
# with parameter rho = 0 (white noise)
# and rho = 0.9 respectively
err1 = rnorm(100)
```


Kraemer W. and Sonnberger H. (1986); *The Linear Regression Model under Test*, Heidelberg: Physica.

Racine J. and Hyndman R. (2002); *Using R To Teach Econometrics*, Journal of Applied Econometrics 17, 175–189.


# Generate Regressor and Dependent Variable
x = rep(c(-1, 1), 50)
y1 = 1 + x + err1
# Perform Durbin-Watson Test:
lmTest(y1 ~ x, "dw")
err2 = filter(err1, 0.9, method = "recursive")
y2 = 1 + x + err2
lmTest(y2 ~ x, "dw")

## gq -
# Generate a Regressor:
x = rep(c(-1, 1), 50)
# Generate Heteroskedastic and Homoskedastic Disturbances:
err1 = c(rnorm(50, sd = 1), rnorm(50, sd = 2))
err2 = rnorm(100)
# Generate a Linear Relationship:
y1 = 1 + x + err1
y2 = 1 + x + err2
# Perform Goldfeld-Quandt Test:
lmTest(y1 ~ x, "gq")
lmTest(y2 ~ x, "gq")

## harv -
# Generate a Regressor and Dependent Variable:
x = 1:50
y1 = 1 + x + rnorm(50)
y2 = y1 + 0.3*x^2
# Perform Harvey-Collier Test:
harv = lmTest(y1 ~ x, "harv")
harv
# Calculate Critical Value for 0.05 level:
qt(0.95, harv$parameter)
lmTest(y2 ~ x, "harv")

## hmc -
# Generate a Regressor:
x = rep(c(-1, 1), 50)
# Generate Heteroskedastic and Homoskedastic Disturbances:
err1 = c(rnorm(50, sd = 1), rnorm(50, sd = 2))
err2 = rnorm(100)
# Generate a Linear Relationship:
y1 = 1 + x + err1
y2 = 1 + x + err2
# Perform Harrison-McCabe Test:
lmTest(y1 ~ x, "hmc")
lmTest(y2 ~ x, "hmc")

## rain -
# Generate Series:
x = c(1:30)
y = x^2 + rnorm(30, 0, 2)
# Perform rainbow Test
rain = lmTest(y ~ x, "rain")
## Example

```r
# Compute Critical Value:
qf(0.95, rain$parameter[1], rain$parameter[2])

## reset -
# Generate Series:
\[
x = c(1:30)
y1 = 1 + x + x^2 + rnorm(30)
y2 = 1 + x + rnorm(30)
\]
# Perform RESET Test:
```
```text
lmTest(y1 ~ x, "reset", power = 2, type = "regressor")
```
```
lmTest(y2 ~ x, "reset", power = 2, type = "regressor")
```

---

### regSim

**Regression Model Simulation**

#### Description

Simulates regression models.

#### Usage

```r
regSim(model = "LM3", n = 100, 
regSim(model = "LOGIT3", n = 100, 
regSim(model = "GAM3", n = 100, 
```

#### Arguments

- `model` (a character string defining the function name from which the regression model will be simulated.)
- `n` (an integer value setting the length, i.e. the number of records of the output series, an integer value. By default n=100.)
- `seed` (an integer value, the recommended way to specify seeds for random number generation.)
- `...` (arguments to be passed to the underlying function specified by the `model` argument.)

#### Details

The function `regSim` allows to simulate from various regression models defined by one of the three example functions `LM3`, `LOGIT3`, `GAM3` or by a user specified function. The examples are defined in the following way:

```r
# LM3:
> y = 0.75 * x1 + 0.25 * x2 - 0.5 * x3 + 0.1 * eps
```
# LOGIT3:
> y = 1 / (1 + exp(- 0.75 * x1 + 0.25 * x2 - 0.5 * x3 + eps))

# GAM3:
> y = scale(scale(sin(2 * pi * x1)) + scale(exp(x2)) + scale(x3))
> y = y + 0.1 * rnorm(n, sd = sd(y))

"LM3" models a linear regression model, "LOGIT3" a generalized linear regression model expressed by a logit model, and "GAM" an additive model. \( x_1, x_2, x_3, \) and \( \text{eps} \) are random normal deviates of length \( n \).

The model function should return an rectangular series defined as an object of class \texttt{data.frame}, \texttt{timeSeries} or \texttt{mts} which can be accepted from the parameter estimation functions \texttt{regFit} and \texttt{gregFit}.

Value

The function \texttt{garchSim} returns an object of the same class as returned by the underlying function \texttt{match.fun(model)}. These may be objects of class \texttt{data.frame}, \texttt{timeSeries} or \texttt{mts}.

Note

This function is still under development. For the future we plan, that the function \texttt{regSim} will be able to generate general regression models.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```r
## LM2 -
# Data for a user defined linear regression model:
LM2 = function(n){
  x = rnorm(n)
  y = rnorm(n)
  eps = 0.1 * rnorm(n)
  z = 0.5 + 0.75 * x + 0.25 * y + eps
  data.frame(Z = z, X = x, Y = y)
}
for (FUN in c("LM2", "LM3")) {
  cat(FUN, ":\n", sep = "")
  print(regSim(model = FUN, n = 10))
}
```
### Extract Regression Model Residuals

**Description**

Extracts residuals from a fitted regression object.

**Usage**

```r
## S4 method for signature 'fREG'
residuals(object)
```

**Arguments**

- **object**: an object of class `fREG` as returned from the function `regFit()` or `gregFit()`.

**Methods**

- `object = "ANY"` Generic function
- `object = "fREG"` Residuals

**Note**

`residuals` is a generic function which extracts residual values from objects returned by modeling functions.

**Author(s)**

Diethelm Wuertz for the Rmetrics R-port.

**Examples**

```r
## regSim -
x = regSim(model = "LM3", n = 50)

## regFit -
fit = regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")

## residuals -
residuals(fit)
```
show-methods

Regression Modelling Show Methods

Description
Show methods for regression modelling.

Details
The show or print method returns the same information for all supported regression models through the use argument in the function \texttt{regFit}.
These are the 'title', the 'formula', the 'family' and the 'model parameters'.

Methods
\begin{verbatim}
object = "ANY" Generic function.
object = "fREG" Print method for objects of class 'fREG'.
\end{verbatim}

Author(s)
Diethelm Wuertz for the Rmetrics R-port.

Examples
\begin{verbatim}
## regSim -
x <- regSim(model = "LM3", n = 50)

## regFit -
fit <- regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")

## print -
print(fit)
\end{verbatim}

summary-methods

Regression Summary Methods

Description
Summary methods for regressing modelling.

Methods
\begin{verbatim}
object = "ANY" Generic function
object = "fREG" Summary method for objects of class 'fREG'.
\end{verbatim}
termPlot

Author(s)
Diethelm Wuertz for the Rmetrics R-port.

Examples
## regSim -
  x <- regSim(model = "LM3", n = 50)

## regFit -
  fit <- regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")

## summary
  summary(fit)

Description
Plots results obtained from a fitted regression model.

Usage
## S3 method for class 'fREG'
termPlot(model, ...)

Arguments
  model an object of class 'fREG'.
  ... additional arguments to be passed to the underlying functions.

Methods
x = "ANY" Generic function.
  x = "fREG" Term plot function.

Author(s)
Diethelm Wuertz for the Rmetrics R-port.

Examples
## regSim -
  x <- regSim(model = "LM3", n = 50)

## regFit -
  fit <- regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")
Regression Model Plot Methods

Description

Plots results obtained from a fitted regression model.

Usage

```r
## S4 method for signature 'fREG'
terms(x, ...)
```

Arguments

- `x`: an object of class `fREG`.
- `...`: additional arguments to be passed to the underlying functions.

Methods

- `x = "ANY"` Generic function.
- `x = "fREG"` Terms extractor function.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```r
## regSim -
x <- regSim(model = "LM3", n = 50)

## regFit -
fit <- regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")
```

Extract Regression Model vcov

Description

Extracts vcov from a fitted regression model.

Methods

- `object = "ANY"` Generic function
- `object = "fREG"` Extractor function for vcov.
Note

vcov is a generic function which extracts fitted values from objects returned by modeling functions, here the `regFit` and `gregFit` parameter estimation functions.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```r
## regSim -
  x <- regSim(model = "LM3", n = 50)

## regFit -
  fit <- regFit(Y ~ X1 + X2 + X3, data = x, use = "lm")

## vcov -
  vcov(fit)
```
Index

* htest
  RegressionTestsInterface, 14
* models
  coef-methods, 4
  fitted-methods, 5
  formula-methods, 6
  plot-methods, 8
  predict-methods, 9
  regFit, 10
  regSim, 22
  residuals-methods, 24
  show-methods, 25
  summary-methods, 25
  termPlot, 26
  terms-methods, 27
  vcov-methods, 27
* package
  fRegression-package, 2
* programming
  fREG-class, 7

bgTest (RegressionTestsInterface), 14
bpTest (RegressionTestsInterface), 14
coeff, ANY-method (coef-methods), 4
coeff, fREG-method (coef-methods), 4
coeff-methods, 4
dwTest (RegressionTestsInterface), 14

family, 10
fitted, ANY-method (fitted-methods), 5
fitted, fREG-method (fitted-methods), 5
fitted-methods, 5
formula, ANY-method (formula-methods), 6
formula, fREG-method (formula-methods), 6
formula-methods, 6
fREG-class, 7
fRegression (fRegression-package), 2
fRegression-package, 2
GAM3 (regSim), 22
glm, 10
gqTest (RegressionTestsInterface), 14
gregFit (regFit), 10
harvTest (RegressionTestsInterface), 14
hmcTest (RegressionTestsInterface), 14
lm, 10
LM3 (regSim), 22
lmTest (RegressionTestsInterface), 14
LOGIT3 (regSim), 22
plot, ANY, ANY-method (plot-methods), 8
plot, fREG, missing-method
  (plot-methods), 8
plot-methods, 8
ppr, 10
predict, ANY-method (predict-methods), 9
predict, fREG-method (predict-methods), 9
predict-methods, 9
rainTest (RegressionTestsInterface), 14
regFit, 10
RegressionTestsInterface, 14
regSim, 22
resetTest (RegressionTestsInterface), 14
residuals, ANY-method
  (residuals-methods), 24
residuals, fREG-method
  (residuals-methods), 24
residuals-methods, 24
show, ANY-method (show-methods), 25
show, fREG-method (show-methods), 25
show-methods, 25
summary, ANY-method (summary-methods), 25
summary, fREG-method (summary-methods), 25
summary-methods, 25
termPlot, 26
terms, ANY-method (terms-methods), 27
terms, fREG-method (terms-methods), 27
terms-methods, 27

vcov, ANY-method (vcov-methods), 27
vcov, fREG-method (vcov-methods), 27
vcov-methods, 27