Package ‘lmtest’

October 13, 2022

Title Testing Linear Regression Models
Version 0.9-40
Date 2022-03-21
Description A collection of tests, data sets, and examples
for diagnostic checking in linear regression models. Furthermore,
some generic tools for inference in parametric models are provided.

LazyData yes
Depends R (>= 3.0.0), stats, zoo
Suggests car, strucchange, sandwich, dynlm, stats4, survival, AER
Imports graphics
License GPL-2 | GPL-3
NeedsCompilation yes

Author Torsten Hothorn [aut] (<https://orcid.org/0000-0001-8301-0471>),
Achim Zeileis [aut, cre] (<https://orcid.org/0000-0003-0918-3766>),
Richard W. Farebrother [aut] (pan.f),
Clint Cummins [aut] (pan.f),
Giovanni Millo [ctb],
David Mitchell [ctb]

Maintainer Achim Zeileis <Achim.Zeileis@R-project.org>

Repository CRAN
Date/Publication 2022-03-21 23:00:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>bptest</td>
<td>2</td>
</tr>
<tr>
<td>bndyield</td>
<td>4</td>
</tr>
<tr>
<td>bptest</td>
<td>6</td>
</tr>
<tr>
<td>ChickEgg</td>
<td>8</td>
</tr>
<tr>
<td>coeftest</td>
<td>9</td>
</tr>
<tr>
<td>coxtest</td>
<td>11</td>
</tr>
<tr>
<td>currenciesubstitution</td>
<td>13</td>
</tr>
</tbody>
</table>
bgtest

Breusch-Godfrey Test

Description

bgtest performs the Breusch-Godfrey test for higher-order serial correlation.

Usage

bgtest(formula, order = 1, order.by = NULL, type = c("Chisq", "F"),
data = list(), fill = 0)

Arguments

- formula: A symbolic description for the model to be tested (or a fitted "lm" object).
- order: Integer. Maximal order of serial correlation to be tested.
- order.by: Either a vector z or a formula with a single explanatory variable like ~ z. The observations in the model are ordered by the size of z. If set to NULL (the default) the observations are assumed to be ordered (e.g., a time series).
- type: The type of test statistic to be returned. Either "Chisq" for the Chi-squared test statistic or "F" for the F test statistic.
- data: An optional data frame containing the variables in the model. By default the variables are taken from the environment which bgtest is called from.
fill starting values for the lagged residuals in the auxiliary regression. By default 0 but can also be set to NA.

Details

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 a finite sample version of the test statistic, employing an $F$ distribution with degrees of freedom as given in parameter.

By default, the starting values for the lagged residuals in the auxiliary regression are chosen to be 0 (as in Godfrey 1978) but could also be set to NA to omit them.

bgtest also returns the coefficients and estimated covariance matrix from the auxiliary regression that includes the lagged residuals. Hence, coefTest can be used to inspect the results. (Note, however, that standard theory does not always apply to the standard errors and t-statistics in this regression.)

Value

A list with class "bgtest" inheriting from "htest" containing the following components:

- **statistic**: the value of the test statistic.
- **p.value**: the p-value of the test.
- **parameter**: degrees of freedom.
- **method**: a character string indicating what type of test was performed.
- **data.name**: a character string giving the name(s) of the data.
- **coefficients**: coefficient estimates from the auxiliary regression.
- **vcov**: corresponding covariance matrix estimate.

Author(s)

David Mitchell <david.mitchell@dotars.gov.au>, Achim Zeileis

References


See Also

dwtest
Examples

```r
## Generate a stationary and an AR(1) series
x <- rep(c(1, -1), 50)
y1 <- 1 + x + rnorm(100)

## Perform Breusch-Godfrey test for first-order serial correlation:
bgtest(y1 ~ x)
## or for fourth-order serial correlation
bgtest(y1 ~ x, order = 4)
## Compare with Durbin-Watson test results:
dwtest(y1 ~ x)

y2 <- filter(y1, 0.5, method = "recursive")
bgtest(y2 ~ x)
bg4 <- bgtest(y2 ~ x, order = 4)
bg4
coefftest(bg4)
```

### bondyield

<table>
<thead>
<tr>
<th>bondyield</th>
<th>Bond Yield</th>
</tr>
</thead>
<tbody>
<tr>
<td>A multivariate quarterly time series from 1961(1) to 1975(4) with variables</td>
<td></td>
</tr>
<tr>
<td>RAARUS difference of interest rate on government and corporate bonds,</td>
<td></td>
</tr>
<tr>
<td>MOOD measure of consumer sentiment,</td>
<td></td>
</tr>
<tr>
<td>EPI index of employment pressure,</td>
<td></td>
</tr>
<tr>
<td>EXP interest rate expectations,</td>
<td></td>
</tr>
<tr>
<td>Y artificial time series based on RAARUS,</td>
<td></td>
</tr>
<tr>
<td>K artificial time series based on RAARUS.</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

Bond Yield Data.

**Usage**

```r
data(bondyield)
```

**Format**

A multivariate quarterly time series from 1961(1) to 1975(4) with variables

RAARUS difference of interest rate on government and corporate bonds,

MOOD measure of consumer sentiment,

EPI index of employment pressure,

EXP interest rate expectations,

Y artificial time series based on RAARUS,

K artificial time series based on RAARUS.

**Source**

The data was originally studied by Cook and Hendershott (1978) and Yawitz and Marshall (1981), the data set is given in Krämer and Sonnberger (1986). Below we replicate a few examples given in their book. Some of these results differ more or less seriously and are sometimes parameterized differently.
References


Examples

```r
data(bondyield)

## page 134, fit Cook-Hendershott OLS model and Yawitz-Marshall OLS model
## third and last line in Table 6.5

modelCH <- RAARUS ~ MOOD + EPI + EXP + RUS
lm(modelCH, data=bondyield)
dwtest(modelCH, data=bondyield)
## wrong sign of RUS coefficient

modelYM <- RAARUS ~ MOOD + Y + K
lm(modelYM, data=bondyield)
dwtest(modelYM, data=bondyield)
## coefficient of Y and K differ by factor 100

## page 135, fit test statistics in Table 6.6 b)

### Chow 1971(1)
if(require(strucchange, quietly = TRUE)) {
sctest(modelCH, point=c(1971,1), data=bondyield, type="Chow")
}

### Breusch-Pagan
bptest(modelCH, data=bondyield, studentize=FALSE)
bptest(modelCH, data=bondyield)

### Fluctuation test
if(require(strucchange, quietly = TRUE)) {
sctest(modelCH, type="fluctuation", data=bondyield, rescale=FALSE)
}

### RESET
reset(modelCH, data=bondyield)
reset(modelCH, power=2, type="regressor", data=bondyield)
reset(modelCH, type="princomp", data=bondyield)

### Harvey-Collier
harvtest(modelCH, order.by= ~ MOOD, data=bondyield)
harvtest(modelCH, order.by= ~ EPI, data=bondyield)
harvtest(modelCH, order.by= ~ EXP, data=bondyield)
harvtest(modelCH, order.by= ~ RUS, data=bondyield)
```
## Rainbow
raintest(modelCH, order.by = "mahalanobis", data=bondyield)

## page 136, fit test statistics in Table 6.6 d)

```
# Chow 1966(1)
if(require(strucchange, quietly = TRUE)) {
sctest(modelYM, point=c(1965,4), data=bondyield, type="Chow") }

# Fluctuation test
if(require(strucchange, quietly = TRUE)) {
sctest(modelYM, type="fluctuation", data=bondyield, rescale=FALSE) }

# RESET
reset(modelYM, data=bondyield)
reset(modelYM, power=2, type="regressor", data=bondyield)
reset(modelYM, type="princomp", data=bondyield)

# Harvey-Collier
harvtest(modelYM, order.by= ~ MOOD, data=bondyield)
harvtest(modelYM, order.by= ~ Y, data=bondyield)
harvtest(modelYM, order.by= ~ K, data=bondyield)

# Rainbow
raintest(modelYM, order.by = "mahalanobis", data=bondyield)
```

---

**bptest**

*Breusch-Pagan Test*

**Description**

Performs the Breusch-Pagan test against heteroskedasticity.

**Usage**

```r
bptest(formula, varformula = NULL, studentize = TRUE, data = list(), weights = NULL)
```

**Arguments**

- **formula**: a symbolic description for the model to be tested (or a fitted "lm" object).
- **varformula**: 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.
- **studentize**: logical. If set to TRUE Koenker’s studentized version of the test statistic will be used.
- **data**: an optional data frame containing the variables in the model. By default the variables are taken from the environment which bptest is called from.
- **weights**: an optional vector of weights to be used in the model.
Details

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.

Examples can not only be found on this page, but also on the help pages of the data sets bondyield, currencysubstitution, growthofmoney, moneydemand, unemployment, wages.

Value

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

- statistic: the value of the test statistic.
- p.value: the p-value of the test.
- parameter: degrees of freedom.
- method: a character string indicating what type of test was performed.
- data.name: a character string giving the name(s) of the data.

References


See Also

- lm
- ncvTest

Examples

```r
## 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
bptest(y1 ~ x)
bptest(y2 ~ x)
```
ChickEgg

Chickens, Eggs, and Causality

Description

US chicken population and egg production.

Usage

data(ChickEgg)

Format

An annual time series from 1930 to 1983 with 2 variables.

- **chicken**: number of chickens (December 1 population of all US chickens excluding commercial broilers).
- **egg**: number of eggs (US egg production in millions of dozens).

Source

The data set was provided by Walter Thurman and made available for R by Roger Koenker. Unfortunately, the data is slightly different than the data analyzed in Thurman & Fisher (1988).

References


Examples

```r
## Which came first: the chicken or the egg?
data(ChickEgg)
## chickens granger-cause eggs?
grangertest(egg ~ chicken, order = 3, data = ChickEgg)
## eggs granger-cause chickens?
grangertest(chicken ~ egg, order = 3, data = ChickEgg)

## To perform the same tests 'by hand', you can use dynlm() andwaldtest():
if(require(dynlm)) {
  ## chickens granger-cause eggs?
  em <- dynlm(egg ~ L(egg, 1) + L(egg, 2) + L(egg, 3), data = ChickEgg)
  em2 <- update(em, . ~ . + L(chicken, 1) + L(chicken, 2) + L(chicken, 3))
  waldtest(em, em2)

  ## eggs granger-cause chickens?
  cm <- dynlm(chicken ~ L(chicken, 1) + L(chicken, 2) + L(chicken, 3), data = ChickEgg)
  cm2 <- update(cm, . ~ . + L(egg, 1) + L(egg, 2) + L(egg, 3))
  waldtest(cm, cm2)
}
```
Inference for Estimated Coefficients

Description

coeffest is a generic function for performing z and (quasi-)t Wald tests of estimated coefficients. coefci computes the corresponding Wald confidence intervals.

Usage

coeffest(x, vcov. = NULL, df = NULL, ...)  

## Default S3 method:  
coeffest(x, vcov. = NULL, df = NULL, ..., save = FALSE)

cofcii(x, parm = NULL, level = 0.95, vcov. = NULL, df = NULL, ...)

Arguments

x an object (for details see below).
vcov. a specification of the covariance matrix of the estimated coefficients. This can be specified as a matrix or as a function yielding a matrix when applied to x.
df the degrees of freedom to be used. If this is a finite positive number a t test with df degrees of freedom is performed. In all other cases, a z test (using a normal approximation) is performed. By default it tries to use x$df.residual and performs a z test if this is NULL.
... further arguments passed to the methods and to vcov. in the default method.
save logical. Should the object x itself be saved as an attribute? (This may be useful for further processing of coefftest objects, e.g., as part of model summaries.)
parm a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
level the confidence level required.

Details

The generic function coefftest currently has a default method (which works in particular for "lm" objects) and dedicated methods for objects of class "glm" (as computed by glm), "mlm" (as computed by lm with multivariate responses), "survreg" (as computed by survreg), and "breakpointsfull" (as computed by breakpoints.formula).

The default method assumes that a coef methods exists, such that coef(x) yields the estimated coefficients.

To specify the corresponding covariance matrix vcov. to be used, there are three possibilities: 1. It is pre-computed and supplied in argument vcov.. 2. A function for extracting the covariance matrix from x is supplied, e.g., sandwich, vcovHC, vcovCL, or vcovHAC from package sandwich.
3. vcov. is set to NULL, then it is assumed that a vcov method exists, such that vcov(x) yields a covariance matrix. Illustrations are provided in the examples below.

The degrees of freedom df determine whether a normal approximation is used or a t distribution with df degrees of freedom. The default method computes df.residual(x) and if this is NULL, 0, or Inf a z test is performed. The method for "glm" objects always uses df = Inf (i.e., a z test).

The corresponding Wald confidence intervals can be computed either by applying coefci to the original model or confint to the output of coeftest. See below for examples.

Finally, nobs and logLik methods are provided which work, provided that there are such methods for the original object x. In that case, "nobs" and "logLik" attributes are stored in the coeftest output so that they can be still queried subsequently. If both methods are available, AIC and BIC can also be applied.

Value

coeftest returns an object of class "coeftest" which is essentially a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p values. Attributes for a "method" label, and the "df" are added along with "nobs" and "logLik" (provided that suitable extractor methods nobs and logLik are available). Optionally, the full object x can be saved in an attribute "object" to facilitate further model summaries based on the coeftest result.

coefci returns a matrix (or vector) with columns giving lower and upper confidence limits for each parameter. These will be labeled as (1-level)/2 and 1 - (1-level)/2 in percent.

See Also

lm, waldtest

Examples

```r
## load data and fit model
data("Mandible", package = "lmtest")
fml <- lm(length ~ age, data = Mandible, subset=(age <= 28))

## the following commands lead to the same tests:
summary(fml)
(ct <- coeftest(fml))

## a z test (instead of a t test) can be performed by
coeftest(fml, df = Inf)

## corresponding confidence intervals
confint(ct)
coefci(fml)
## which in this simple case is equivalent to
confint(fml)

## extract further model information either from
## the original model or from the coeftest output
nobs(fml)
nobs(ct)
logLik(fml)
```
if(require("sandwich")) {
  ## a different covariance matrix can be also used:
  (ct <- coeftest(fm, df = Inf, vcov = vcovHC))

  ## the corresponding confidence interval can be computed either as
  confint(ct)
  ## or based on the original model
  coefci(fm, df = Inf, vcov = vcovHC)

  ## note that the degrees of freedom _actually used_ can be extracted
  df.residual(ct)
  ## which differ here from
  df.residual(fm)

  ## vcov can also be supplied as a function with additional arguments
  coeftest(fm, df = Inf, vcov = vcovHC, type = "HC0")
  ## or as a matrix
  coeftest(fm, df = Inf, vcov = vcovHC(fm, type = "HC0"))
}

---

**coxtest**  
*Cox Test for Comparing Non-Nested Models*

**Description**

*coxtest* performs the Cox test for comparing two non-nested models.

**Usage**

```
coxtest(formula1, formula2, data = list())
```

**Arguments**

- **formula1**: either a symbolic description for the first model to be tested, or a fitted object of class “lm”.
- **formula2**: either a symbolic description for the second model to be tested, or a fitted object of class “lm”.
- **data**: an optional data frame containing the variables in the model. By default the variables are taken from the environment which *coxtest* is called from.
Details

The idea of the Cox test is the following: if the first model contains the correct set of regressors, then a fit of the regressors from the second model to the fitted values from first model should have no further explanatory value. But if it has, it can be concluded that model 1 does not contain the correct set of regressors.

Hence, to compare both models the fitted values of model 1 are regressed on model 2 and vice versa. A Cox test statistic is computed for each auxiliary model which is asymptotically standard normally distributed.

For further details, see the references.

Value

An object of class "anova" which contains the estimate plus corresponding standard error, z test statistic and p value for each auxiliary test.

References


See Also

jtest, encomptest

Examples

```r
## Fit two competing, non-nested models for aggregate consumption, as in Greene (1993), Examples 7.11 and 7.12

## load data and compute lags
data(USDistLag)
usdl <- na.contiguous(cbind(USDistLag, lag(USDistLag, k = -1)))
colnames(usdl) <- c("con", "gnp", "con1", "gnp1")

## C(t) = a0 + a1*Y(t) + a2*C(t-1) + u
fm1 <- lm(con ~ gnp + con1, data = usdl)

## C(t) = b0 + b1*Y(t) + b2*Y(t-1) + v
fm2 <- lm(con ~ gnp + gnp1, data = usdl)

## Cox test in both directions:
coxtest(fm1, fm2)

## ...and do the same for jtest() and encomptest().
## Notice that in this particular case they are coincident.
jtest(fm1, fm2)
encomptest(fm1, fm2)
```
Description
Currency Substitution Data.

Usage
data(currencysubstitution)

Format
A multivariate quarterly time series from 1960(4) to 1975(4) with variables

- logCUS logarithm of the ratio of Canadian holdings of Canadian dollar balances and Canadian holdings of U.S. dollar balances,
- Iu yield on U.S. Treasury bills,
- Ic yield on Canadian Treasury bills,
- logY logarithm of Canadian real gross national product.

Source
The data was originally studied by Miles (1978), the data set is given in Krämer and Sonnberger (1986). Below we replicate a few examples from their book. Some of these results differ more or less seriously and are sometimes parameterized differently.

References

Examples
data(currencysubstitution)

## page 130, fit Miles OLS model and Bordo-Choudri OLS model
## third and last line in Table 6.3
modelMiles <- logCUS ~ log((1+Iu)/(1+Ic))
lm(modelMiles, data=currencysubstitution)
dwtest(modelMiles, data=currencysubstitution)

modelBordoChoudri <- logCUS ~ I(Iu-Ic) + Ic + logY
lm(modelBordoChoudri, data=currencysubstitution)
dwtest(modelBordoChoudri, data=currencysubstitution)
if(require(strucchange, quietly = TRUE)) {
  ## Fluctuation test
  sctest(modelMiles, type="fluctuation", data=currencysubstitution,
          rescale=FALSE) }

## RESET
reset(modelMiles, data=currencysubstitution)
reset(modelMiles, power=2, type="regressor", data=currencysubstitution)
reset(modelMiles, type="princomp", data=currencysubstitution)

## Harvey-Collier
harvtest(modelMiles, order.by = ~log((1+Iu)/(1+Ic)), data=currencysubstitution)

## Rainbow
raintest(modelMiles, order.by = "mahalanobis", data=currencysubstitution)

if(require(strucchange, quietly = TRUE)) {
  ## Chow 1970(2)
  sctest(modelBordoChoudri, point=c(1970,2), data=currencysubstitution,
          type="Chow") }

## Breusch-Pagan
bptest(modelBordoChoudri, data=currencysubstitution, studentize=FALSE)
bptest(modelBordoChoudri, data=currencysubstitution)

## RESET
reset(modelBordoChoudri, data=currencysubstitution)
reset(modelBordoChoudri, power=2, type="regressor", data=currencysubstitution)
reset(modelBordoChoudri, type="princomp", data=currencysubstitution)

## Harvey-Collier
harvtest(modelBordoChoudri, order.by = ~I(Iu-Ic), data=currencysubstitution)
harvtest(modelBordoChoudri, order.by = ~Ic, data=currencysubstitution)
harvtest(modelBordoChoudri, order.by = ~logY, data=currencysubstitution)

## Rainbow
raintest(modelBordoChoudri, order.by = "mahalanobis", data=currencysubstitution)

---

**dwtest**

**Durbin-Watson Test**

**Description**

Performs the Durbin-Watson test for autocorrelation of disturbances.
Usage

dwtest(formula, order.by = NULL, alternative = c("greater", "two.sided", "less"),
       iterations = 15, exact = NULL, tol = 1e-10, data = list())

Arguments

- **formula**: a symbolic description for the model to be tested (or a fitted "lm" object).
- **order.by**: Either a vector \( z \) or a formula with a single explanatory variable like \( \sim z \). The observations in the model are ordered by the size of \( z \). If set to NULL (the default) the observations are assumed to be ordered (e.g., a time series).
- **alternative**: a character string specifying the alternative hypothesis.
- **iterations**: an integer specifying the number of iterations when calculating the p-value with the "pan" algorithm.
- **exact**: logical. 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.
- **tol**: tolerance. Eigenvalues computed have to be greater than \( \text{tol} \) to be treated as non-zero.
- **data**: an optional data frame containing the variables in the model. By default the variables are taken from the environment which dwtest is called from.

Details

The Durbin-Watson test has the null hypothesis that the autocorrelation of the disturbances is 0. It is possible to test against the alternative that it is greater than, not equal to, or less than 0, respectively. This can be specified by the alternative argument.

Under the assumption of normally distributed disturbances, the null distribution of the Durbin-Watson statistic is the distribution of a linear combination of chi-squared variables. The p-value is computed using the Fortran version of 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.

Examples can not only be found on this page, but also on the help pages of the data sets bondyield, currencysubstitution, growthofmoney, moneydemand, unemployment, wages.

Value

An object of class "htest" containing:

- **statistic**: the test statistic.
- **method**: a character string with the method used.
- **alternative**: a character string describing the alternative hypothesis.
- **p.value**: the corresponding p-value.
- **data.name**: a character string with the data name.
References


See Also

`lm`

Examples

```r
## generate two AR(1) error terms with parameter
## rho = 0 (white noise) and rho = 0.9 respectively
err1 <- rnorm(100)

## generate regressor and dependent variable
x <- rep(c(-1,1), 50)
y1 <- 1 + x + err1

dwtest(y1 ~ x)

err2 <- filter(err1, 0.9, method="recursive")
y2 <- 1 + x + err2
dwtest(y2 ~ x)
```

**encomptest**

*Encompassing Test for Comparing Non-Nested Models*

Description

encomptest performs the encompassing test of Davidson & MacKinnon for comparing non-nested models.
encomptest

Usage

encomptest(formula1, formula2, data = list(), vcov. = NULL, ...)

Arguments

formula1  either a symbolic description for the first model to be tested, or a fitted object of class "lm".

formula2  either a symbolic description for the second model to be tested, or a fitted object of class "lm".

data  an optional data frame containing the variables in the model. By default the variables are taken from the environment which encomptest is called from.

vcov.  a function for estimating the covariance matrix of the regression coefficients, e.g., vcovHC.

...  further arguments passed to waldtest.

Details

To compare two non-nested models, the encompassing test fits an encompassing model which contains all regressors from both models such that the two models are nested within the encompassing model. A Wald test for comparing each of the models with the encompassing model is carried out by waldtest.

For further details, see the references.

Value

An object of class "anova" which contains the residual degrees of freedom in the encompassing model, the difference in degrees of freedom, Wald statistic (either "F" or "Chisq") and corresponding p value.

References


See Also

coxtest, jtest

Examples

## Fit two competing, non-nested models for aggregate consumption, as in Greene (1993), Examples 7.11 and 7.12
## load data and compute lags
data(USDistLag)
usdl <- na.contiguous(cbind(USDistLag, lag(USDistLag, k = -1)))
colnames(usdl) <- c("con", "gnp", "con1", "gnp1")

## C(t) = a0 + a1*Y(t) + a2*C(t-1) + u
fm1 <- lm(con ~ gnp + con1, data = usdl)

## C(t) = b0 + b1*Y(t) + b2*Y(t-1) + v
fm2 <- lm(con ~ gnp + gnp1, data = usdl)

## Encompassing model
fm3 <- lm(con ~ gnp + con1 + gnp1, data = usdl)

## Cox test in both directions:
coxtest(fm1, fm2)

## ...and do the same for jtest() and encomptest().
## Notice that in this particular case they are coincident.
jtest(fm1, fm2)
encomptest(fm1, fm2)

## the encompassing test is essentially
waldtest(fm1, fm3, fm2)

---

**ftemp**

*Femal Temperature Data*

**Description**

Daily morning temperature of adult female (in degrees Celsius).

**Usage**

data(ftemp)

**Format**

Univariate daily time series of 60 observations starting from 1990-07-11.

**Details**

The data gives the daily morning temperature of an adult woman measured in degrees Celsius at about 6.30am each morning.

At the start of the period the woman was sick, hence the high temperature. Then the usual monthly cycle can be seen. On the second cycle, the temperature doesn’t complete the downward part of the pattern due to a conception.

**Source**

The data set was taken from the Time Series Data Library, maintained by Rob Hyndman.
Examples

data(ftemp)
plot(ftemp)
y <- window(ftemp, start = 8, end = 60)
if(require(strucchange)) {
  bp <- breakpoints(y ~ 1)
  plot(bp)
  fm.seg <- lm(y ~ 0 + breakfactor(bp))
  plot(y)
  lines(8:60, fitted(fm.seg), col = 4)
  lines(confint(bp))
}

---

gqtest  \hspace{2cm} \textit{Goldfeld-Quandt Test}

Description

Goldfeld-Quandt test against heteroskedasticity.

Usage

gqtest(formula, point = 0.5, fraction = 0,
        alternative = c("greater", "two.sided", "less"),
        order.by = NULL, data = list())

Arguments

- formula: a symbolic description for the model to be tested (or a fitted "lm" object).
- point: numerical. If point is smaller than 1 it is interpreted as percentages of data, i.e. \( n \times \text{point} \) 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.
- fraction: numerical. The number of central observations to be omitted. If fraction is smaller than 1, it is chosen to be \( \text{fraction} \times n \) if \( n \) is the number of observations in the model.
- alternative: a character string specifying the alternative hypothesis. The default is to test for increasing variances.
- order.by: Either a vector \( z \) or a formula with a single explanatory variable like \( \sim z \). The observations in the model are ordered by the size of \( z \). If set to NULL (the default) the observations are assumed to be ordered (e.g., a time series).
- data: an optional data frame containing the variables in the model. By default the variables are taken from the environment which gqtest is called from.
Details

The Goldfeld-Quandt test compares the variances of two submodels divided by a specified break-point 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.

Examples can not only be found on this page, but also on the help pages of the data sets bondyield, currencysubstitution, growthofmoney, moneydemand, unemployment, wages.

Value

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

- statistic: the value of the test statistic.
- parameter: degrees of freedom.
- method: a character string indicating what type of test was performed.
- alternative: a character string describing the alternative hypothesis.
- p.value: the p-value of the test.
- data.name: a character string giving the name(s) of the data.

References


See Also

- lm

Examples

```r
## 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
gqtest(y1 ~ x)
gqtest(y2 ~ x)
```
grangertest

Test for Granger Causality

Description

`grangertest` is a generic function for performing a test for Granger causality.

Usage

```r
## Default S3 method:
grangertest(x, y, order = 1, na.action = na.omit, ...)
## S3 method for class 'formula'
grangertest(formula, data = list(), ...)
```

Arguments

- `x` either a bivariate series (in which case `y` has to be missing) or a univariate series of observations.
- `y` a univariate series of observations (if `x` is univariate, too).
- `order` integer specifying the order of lags to include in the auxiliary regression.
- `na.action` a function for eliminating NAs after aligning the series `x` and `y`.
- `...` further arguments passed to `waldtest`.
- `formula` a formula specification of a bivariate series like `y ~ x`.
- `data` an optional data frame containing the variables in the model. By default the variables are taken from the environment which `grangertest` is called from.

Details

Currently, the methods for the generic function `grangertest` only perform tests for Granger causality in bivariate series. The test is simply a Wald test comparing the unrestricted model—in which `y` is explained by the lags (up to order `order`) of `y` and `x`—and the restricted model—in which `y` is only explained by the lags of `y`.

Both methods are simply convenience interfaces to `waldtest`.

Value

An object of class "anova" which contains the residual degrees of freedom, the difference in degrees of freedom, Wald statistic and corresponding p value.

See Also

`waldtest`, `ChickEgg`
### Examples

```r
## Which came first: the chicken or the egg?
data(ChickEgg)
grangertest(egg ~ chicken, order = 3, data = ChickEgg)
grangertest(chicken ~ egg, order = 3, data = ChickEgg)

## alternative ways of specifying the same test
grangertest(ChickEgg[, 1], ChickEgg[, 2], order = 3)
```

#### growthofmoney

**Growth of Money Supply**

**Description**

Growth of Money Supply Data.

**Usage**

`data(growthofmoney)`

**Format**

A multivariate quarterly time series from 1970(2) to 1974(4) with variables

- `TG1.TG0` difference of current and preceding target for the growth rate of the money supply,
- `AG0.TG0` difference of actual growth rate and target growth rate for the preceding period.

**Source**

The data was originally studied by Hetzel (1981), the data set is given in Krämer and Sonnberger (1986). Below we replicate a few examples from their book. Some of these results differ more or less seriously and are sometimes parameterized differently.

**References**


**Examples**

```r
data(growthofmoney)

## page 137, fit Hetzel OLS model
## first/second line in Table 6.7
modelHetzel <- TG1.TG0 ~ AG0.TG0
```
harvtest

\begin{verbatim}
  lm(modelHetzel, data=growthofmoney)
dwtest(modelHetzel, data=growthofmoney)

  ## page 135, fit test statistics in Table 6.8
  #################################################################
  if(require(strucchange, quietly = TRUE)) {
    ## Chow 1974(1)
    sctest(modelHetzel, point=c(1973,4), data=growthofmoney, type="Chow") }

  ## RESET
  reset(modelHetzel, data=growthofmoney)
  reset(modelHetzel, power=2, type="regressor", data=growthofmoney)
  reset(modelHetzel, type="princomp", data=growthofmoney)

  ## Harvey-Collier
  harvtest(modelHetzel, order.by= ~ AG0.TG0, data=growthofmoney)

  ## Rainbow
  raintest(modelHetzel, order.by = "mahalanobis", data=growthofmoney)

  ## Identification of outliers
  #################################################################
  ## Figure 6.1
  plot(modelHetzel, data=growthofmoney)
  abline(v=0)
  abline(h=0)
  abline(coef(lm(modelHetzel, data=growthofmoney)), col=2)

  ## Table 6.7, last line
  growthofmoney2 <- as.data.frame(growthofmoney[-c(5:6),])
  lm(modelHetzel, data=growthofmoney2)
  dwtest(modelHetzel, data=growthofmoney2)
\end{verbatim}

\begin{center}
\begin{tabular}{ll}
\hline
\textit{harvtest} & \textit{Harvey-Collier Test} \\
\hline
\end{tabular}
\end{center}

\textbf{Description}

Harvey-Collier test for linearity.

\textbf{Usage}

\begin{verbatim}
  harvtest(formula, order.by = NULL, data = list())
\end{verbatim}
Arguments

- formula: a symbolic description for the model to be tested (or a fitted "lm" object).
- order.by: Either a vector \( z \) or a formula with a single explanatory variable like \( \sim z \). The observations in the model are ordered by the size of \( z \). If set to \texttt{NULL} (the default) the observations are assumed to be ordered (e.g., a time series).
- data: an optional data frame containing the variables in the model. By default the variables are taken from the environment which \texttt{harvtest} is called from.

Details

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.

Examples can not only be found on this page, but also on the help pages of the data sets \texttt{bondyield}, \texttt{currencysubstitution}, \texttt{growthofmoney}, \texttt{moneydemand}, \texttt{unemployment}, \texttt{wages}.

Value

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

- statistic: the value of the test statistic.
- p.value: the p-value of the test.
- parameter: degrees of freedom.
- method: a character string indicating what type of test was performed.
- data.name: a character string giving the name(s) of the data.

References


See Also

\texttt{lm}

Examples

```r
# 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 <- harvtest(y1 ~ x)
harv

## calculate critical value vor 0.05 level
qt(0.95, harv$parameter)
harvtest(y2 ~ x)
```
hmctest

**Harrison-McCabe test**

**Description**

Harrison-McCabe test for heteroskedasticity.

**Usage**

```
hmctest(formula, point = 0.5, order.by = NULL, simulate.p = TRUE, nsim = 1000,
         plot = FALSE, data = list())
```

**Arguments**

- `formula`: a symbolic description for the model to be tested (or a fitted "lm" object).
- `point`: numeric. 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.
- `order.by`: Either a vector `z` or a formula with a single explanatory variable like `~ z`. The observations in the model are ordered by the size of `z`. If set to `NULL` (the default) the observations are assumed to be ordered (e.g., a time series).
- `simulate.p`: logical. If `TRUE` a p value will be assessed by simulation, otherwise the p value is NA.
- `nsim`: integer. Determines how many runs are used to simulate the p value.
- `plot`: logical. If `TRUE` the test statistic for all possible breakpoints is plotted.
- `data`: an optional data frame containing the variables in the model. By default the variables are taken from the environment which `hmctest` is called from.

**Details**

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.

Examples can not only be found on this page, but also on the help pages of the data sets `bondyield`, `currencysubstitution`, `growthofmoney`, `moneydemand`, `unemployment`, `wages`.

**Value**

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

- `statistic`: the value of the test statistic.
- `p.value`: the simulated p-value of the test.
- `method`: a character string indicating what type of test was performed.
- `data.name`: a character string giving the name(s) of the data.
## References


### See Also

- `lm`

### Examples

```r
## 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
hmctest(y1 ~ x)
hmctest(y2 ~ x)
```

---

### Description

Several macroeconomic time series from the U.S.

### Usage

```r
data(fyff)
data(gmdc)
data(ip)
data(jocci)
data(lhur)
data(pw561)
```

### Format

All data sets are multivariate monthly time series from 1959(8) to 1993(12) (except 1993(10) for `jocci`) with variables

- **y** original time series,
- **dy** transformed times series (first differences or log first differences),
- **dy1** transformed series at lag 1,
**Details**

The description from Stock & Watson (1996) for the time series (with the transformation used):

- **fyff** interest rate (first differences),
- **gmdc** pce, implicit price deflator: pce (1987 = 100) (log first differences),
- **ip** index of industrial production (log first differences),
- **jocci** department of commerce commodity price index (log first differences),
- **lhur** unemployment rate: all workers, 16 years & over (%, sa) (first differences),
- **pw561** producer price index: crude petroleum (82 = 100, nsa) (log first differences).

Stock & Watson (1996) fitted an AR(6) model to all transformed time series.

**Source**


**References**


**Examples**

```r
data(jocci)

dwtest(dy ~ 1, data = jocci)
bgtest(dy ~ 1, data = jocci)
ar6.model <- dy ~ dy1 + dy2 + dy3 + dy4 + dy5 + dy6
bgtest(ar6.model, data = jocci)

var.model <- ~ I(dy1^2) + I(dy2^2) + I(dy3^2) + I(dy4^2) + I(dy5^2) + I(dy6^2)
bptest(ar6.model, var.model, data = jocci)
```
Description

jtest performs the Davidson-MacKinnon J test for comparing non-nested models.

Usage

jtest(formula1, formula2, data = list(), vcov. = NULL, ...)

Arguments

formula1 either a symbolic description for the first model to be tested, or a fitted object of class "lm".
formula2 either a symbolic description for the second model to be tested, or a fitted object of class "lm".
data an optional data frame containing the variables in the model. By default the variables are taken from the environment which jtest is called from.
vcov. a function for estimating the covariance matrix of the regression coefficients, e.g., vcovHC.
... further arguments passed to coeftest.

Details

The idea of the J test is the following: if the first model contains the correct set of regressors, then including the fitted values of the second model into the set of regressors should provide no significant improvement. But if it does, it can be concluded that model 1 does not contain the correct set of regressors.

Hence, to compare both models the fitted values of model 1 are included into model 2 and vice versa. The J test statistic is simply the marginal test of the fitted values in the augmented model. This is performed by coeftest.

For further details, see the references.

Value

An object of class "anova" which contains the coefficient estimate of the fitted values in the augmented regression plus corresponding standard error, test statistic and p value.

References


See Also

coxtest, encomptest

Examples

```r
## Fit two competing, non-nested models for aggregate
## consumption, as in Greene (1993), Examples 7.11 and 7.12

## load data and compute lags
data(USDistLag)
usdl <- na.contiguous(cbind(USDistLag, lag(USDistLag, k = -1)))
colnames(usdl) <- c("con", "gnp", "con1", "gnp1")

## C(t) = a0 + a1*Y(t) + a2*C(t-1) + u
fm1 <- lm(con ~ gnp + con1, data = usdl)

## C(t) = b0 + b1*Y(t) + b2*Y(t-1) + v
fm2 <- lm(con ~ gnp + gnp1, data = usdl)

## Cox test in both directions:
coxtest(fm1, fm2)

## ...and do the same for jtest() and encomptest().
## Notice that in this particular case they are coincident.
jtest(fm1, fm2)
encomptest(fm1, fm2)
```

---

**lrtest**

*Likelihood Ratio Test of Nested Models*

**Description**

`lrtest` is a generic function for carrying out likelihood ratio tests. The default method can be employed for comparing nested (generalized) linear models (see details below).

**Usage**

```r
lrtest(object, ...)  

## Default S3 method:
lrtest(object, ..., name = NULL)

## S3 method for class 'formula'
lrtest(object, ..., data = list())
```
**Arguments**

- **object**: an object. See below for details.
- **...**: further object specifications passed to methods. See below for details.
- **name**: a function for extracting a suitable name/description from a fitted model object. By default the name is queried by calling `formula`.
- **data**: a data frame containing the variables in the model.

**Details**

`lrtest` is intended to be a generic function for comparisons of models via asymptotic likelihood ratio tests. The default method consecutively compares the fitted model object `object` with the models passed in `...`. Instead of passing the fitted model objects in `...`, several other specifications are possible. The updating mechanism is the same as for `waldtest`: the models in `...` can be specified as integers, characters (both for terms that should be eliminated from the previous model), update formulas or fitted model objects. Except for the last case, the existence of an `update` method is assumed. See `waldtest` for details.

Subsequently, an asymptotic likelihood ratio test for each two consecutive models is carried out: Twice the difference in log-likelihoods (as derived by the `logLik` methods) is compared with a Chi-squared distribution.

The "formula" method fits a `lm` first and then calls the default method.

**Value**

An object of class "anova" which contains the log-likelihood, degrees of freedom, the difference in degrees of freedom, likelihood ratio Chi-squared statistic and corresponding p value.

**See Also**

- `waldtest`

**Examples**

```r
## with data from Greene (1993):
## load data and compute lags
data("USDistLag")
usdl <- na.contiguous(cbind(USDistLag, lag(USDistLag, k = -1)))
colnames(usdl) <- c("con", "gnp", "con1", "gnp1")

fm1 <- lm(con ~ gnp + gnp1, data = usdl)
fm2 <- lm(con ~ gnp + con1 + gnp1, data = usdl)

## various equivalent specifications of the LR test
lrtest(fm2, fm1)
lrtest(fm2, 2)
lrtest(fm2, "con1")
lrtest(fm2, . ~ . - con1)
```
Mandible

Description
Mandible Data.

Usage
data(Mandible)

Format
Data from 167 fetuses, especially:

- **age** gestational age in weeks.
- **length** mandible length in mm.

Source
The data was originally published by Chitty et al., 1993, and analyzed in Royston and Altman, 1994 (the data is given there). Only measurements with age <= 28 were used in this analysis.

References

Examples
data(Mandible)
lm(length ~ age, data=Mandible, subset=(age <= 28))

moneydemand

Description
Money Demand Data.

Usage
data(moneydemand)
A multivariate yearly time series from 1879 to 1974 with variables

- $\log M$ logarithm of quantity of money,
- $\log Y_p$ logarithm of real permanent income,
- $R_s$ short term interest rate,
- $R_m$ rate of return on money,
- $R_l$ not documented in the sources,
- $\log S_{pp}$ logarithm of an operational measure of the variability of the rate of price changes.

The data was originally studied by Allen (1982), the data set is given in Krämer and Sonnberger (1986). Below we replicate a few examples from the book. Some of these results differ more or less seriously and are sometimes parameterized differently.


```r
data(moneydemand)
moneydemand <- window(moneydemand, start=1880, end=1972)

## page 125, fit Allen OLS model (and Durbin-Watson test),
## last line in Table 6.1
modelAllen <- logM ~ logYp + Rs + Rl + logSpp
lm(modelAllen, data = moneydemand)
dwtest(modelAllen, data = moneydemand)

## page 127, fit test statistics in Table 6.1 c)
################################################
## Breusch-Pagan
bptest(modelAllen, studentize = FALSE, data = moneydemand)
bptest(modelAllen, studentize = TRUE, data = moneydemand)

## RESET
reset(modelAllen, data = moneydemand)
reset(modelAllen, power = 2, type = "regressor", data = moneydemand)
reset(modelAllen, type = "princomp", data = moneydemand)

## Harvey-Collier tests (up to sign of the test statistic)
harvtest(modelAllen, order.by = ~logYp, data = moneydemand)
harvtest(modelAllen, order.by = ~Rs, data = moneydemand)
```
```r
harvtest(modelAllen, order.by = ~Rl, data = moneydemand)
harvtest(modelAllen, order.by = ~logSpp, data = moneydemand)

## Rainbow test
raintest(modelAllen, order.by = "mahalanobis", data = moneydemand)

if(require(strucchange, quietly = TRUE)) {
  ## Chow (1913)
sctest(modelAllen, point=c(1913,1), data = moneydemand, type = "Chow")
}
if(require(strucchange, quietly = TRUE)) {
  ## Fluctuation
sctest(modelAllen, type = "fluctuation", rescale = FALSE, data = moneydemand)
}
```

---

**petest**  
*PE Test for Linear vs. Log-Linear Specifications*

**Description**

`petest` performs the MacKinnon-White-Davidson PE test for comparing linear vs. log-linear specifications in linear regressions.

**Usage**

`petest(formula1, formula2, data = list(), vcov. = NULL, ...)`

**Arguments**

- `formula1`: either a symbolic description for the first model to be tested, or a fitted object of class "lm".
- `formula2`: either a symbolic description for the second model to be tested, or a fitted object of class "lm".
- `data`: an optional data frame containing the variables in the model. By default the variables are taken from the environment which `petest` is called from.
- `vcov.`: a function for estimating the covariance matrix of the regression coefficients, e.g., `vcovHC`.
- `...`: further arguments passed to `coeftest`.

**Details**

The PE test compares two non-nest models where one has a linear specification of type `y ~ x1 + x2` and the other has a log-linear specification of type `log(y) ~ z1 + z2`. Typically, the regressors in the latter model are logs of the regressors in the former, i.e., `z1` is `log(x1)` etc.

The idea of the PE test is the following: If the linear specification is correct then adding an auxiliary regressor with the difference of the log-fitted values from both models should be non-significant. Conversely, if the log-linear specification is correct then adding an auxiliary regressor with the
difference of fitted values in levels should be non-significant. The PE test statistic is simply the marginal test of the auxiliary variable(s) in the augmented model(s). In petest this is performed by \texttt{coeftest}.

For further details, see the references.

Value

An object of class "anova" which contains the coefficient estimate of the auxiliary variables in the augmented regression plus corresponding standard error, test statistic and p value.

References


See Also

\texttt{jtest, coxtest, encomptest}

Examples

\begin{verbatim}
if(require("AER")) {
  ## Verbeek (2004), Section 3
  data("HousePrices", package = "AER")

  ### Verbeek (2004), Table 3.3
  hp_lin <- lm(price ~ . , data = HousePrices)
  summary(hp_lin)

  ### Verbeek (2004), Table 3.2
  hp_log <- update(hp_lin, log(price) ~ . - lotsize + log(lotsize))
  summary(hp_log)

  ## PE test
  petest(hp_lin, hp_log)

  ## Greene (2003), Example 9.8
  data("USMacroG", package = "AER")

  ## Greene (2003), Table 9.2
  usm_lin <- lm(m1 ~ tbill + gdp, data = USMacroG)
  usm_log <- lm(log(m1) ~ log(tbill) + log(gdp), data = USMacroG)
  petest(usm_lin, usm_log)
  ## matches results from Greene's errata
}
\end{verbatim}
Description

Rainbow test for linearity.

Usage

raintest(formula, fraction = 0.5, order.by = NULL, center = NULL,
  data=list())

Arguments

- **formula**: a symbolic description for the model to be tested (or a fitted "lm" object).
- **fraction**: numeric. The percentage of observations in the subset is determined by fraction*n if n is the number of observations in the model.
- **order.by**: Either a vector z or a formula with a single explanatory variable like ~ z. The observations in the model are ordered by the size of z. 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 distances from the mean regressor.
- **center**: numeric. 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 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 raintest is called from.

Details

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 worse than the fit for the subsample. The test statistic under $H_0$ follows an F distribution with parameter degrees of freedom.

Examples can not only be found on this page, but also on the help pages of the data sets bondyield, currenciesubstitution, growthofmoney, moneydemand, unemployment, wages.

Value

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

- **statistic**: the value of the test statistic.
p.value  the p-value of the test.
parameter degrees of freedom.
method  a character string indicating what type of test was performed.
data.name  a character string giving the name(s) of the data.

References


See Also

`lm`

Examples

```r
x <- c(1:30)
y <- x^2 + rnorm(30,0,2)
rain <- raintest(y ~ x)
rain
## critical value
qf(0.95, rain$parameter[1], rain$parameter[2])
```

`resettest`  *RESET Test*

Description

Ramsey’s RESET test for functional form.

Usage

```r
resettest(formula, power = 2:3, type = c("fitted", "regressor", "princomp"), data = list(), vcov = NULL, ...)
```

Arguments

- `formula`  a symbolic description for the model to be tested (or a fitted “lm” object).
- `power`  integers. A vector of positive integers indicating the powers of the variables that should be included. By default, the test is for quadratic or cubic influence of the fitted response.
- `type`  a string indicating whether powers of the fitted response, the regressor variables (factors are left out), or the first principal component of the regressor matrix should be included in the extended model.
- `data`  an optional data frame containing the variables in the model. By default the variables are taken from the environment which `resettest` is called from.
- `vcov, ...`  optional arguments to be passed to `waldtest` for carrying out the F test.
Details

The RESET test is a popular diagnostic for correctness of functional form. The basic assumption is that under the alternative the model can be written in the form \( y = X\beta + Z\gamma + u \). \( Z \) 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.

This function was called reset in previous versions of the package. Please use resettest instead.

Examples can not only be found on this page, but also on the help pages of the data sets bondyield, currenciesubstitution, growthofmoney, moneydemand, unemployment, wages.

Value

An object of class "htest" containing:

- statistic: the test statistic.
- p.value: the corresponding p-value.
- parameter: degrees of freedom.
- method: a character string with the method used.
- data.name: a character string with the data name.

References


See Also

lm

Examples

```r
x <- c(1:30)
y1 <- 1 + x + x^2 + rnorm(30)
y2 <- 1 + x + rnorm(30)
resettest(y1 ~ x, power=2, type="regressor")
resettest(y2 ~ x, power=2, type="regressor")
```
Unemployment Data

Description
Unemployment Data.

Usage
data(unemployment)

Format
A multivariate yearly time series from 1890 to 1979 with variables

- UN unemployment rate,
- m broad money supply,
- p implicit deflator of Gross National Product,
- G real purchases of goods and services,
- x real exports.

Source
The data was originally studied by Rea (1983), the data set is given in Krämer and Sonnberger (1986). Below we replicate a few examples from their book. Some of these results differ more or less seriously and are sometimes parameterized differently.

References

Examples
data(unemployment)

## data transformation
myunemployment <- window(unemployment, start=1895, end=1956)
time <- 6:67

## page 144, fit Rea OLS model
## last line in Table 6.12
modelRea <- UN ~ log(m/p) + log(G) + log(x) + time
lm(modelRea, data = myunemployment)
## coefficients of logged variables differ by factor 100
## page 143, fit test statistics in table 6.11

```r
if(require(strucchange, quietly = TRUE)) {
  ## Chow 1941
  sctest(modelRea, point=c(1940,1), data=myunemployment, type="Chow")
}

## Breusch-Pagan
bptest(modelRea, data=myunemployment, studentize=FALSE)
bptest(modelRea, data=myunemployment)

## RESET (a)-(b)
reset(modelRea, data=myunemployment)
reset(modelRea, power=2, type="regressor", data=myunemployment)

## Harvey-Collier
harvtest(modelRea, order.by = ~ log(m/p), data=myunemployment)
harvtest(modelRea, order.by = ~ log(G), data=myunemployment)
harvtest(modelRea, order.by = ~ log(x), data=myunemployment)
harvtest(modelRea, data=myunemployment)

## Rainbow
raintest(modelRea, order.by = "mahalanobis", data=myunemployment)
```

---

**USDistLag**

*US Macroeconomic Data*

**Description**

US macroeconomic data for fitting a distributed lag model.

**Usage**

```r
data(USDistLag)
```

**Format**

An annual time series from 1963 to 1982 with 2 variables.

- **consumption** real consumption,
- **gnp** gross national product (deflated by CPI).

**Source**

Table 7.7 in Greene (1993)
References


Examples

```r
## Willam H. Greene, Econometric Analysis, 2nd Ed.
## Chapter 7
## load data set, p. 221, Table 7.7
data(USDistLag)

## fit distributed lag model, p.221, Example 7.8
usdl <- na.contiguous(cbind(USDistLag, lag(USDistLag, k = -1)))
colnames(usdl) <- c("con", "gnp", "con1", "gnp1")

## C(t) = b0 + b1*Y(t) + b2*C(t-1) + u
fm <- lm(con ~ gnp + con1, data = usdl)
summary(fm)
vcov(fm)
```

valueofstocks

<table>
<thead>
<tr>
<th>Description</th>
<th>Value of Stocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of Stocks Data</td>
<td></td>
</tr>
</tbody>
</table>

Usage

data(valueofstocks)

Format

A multivariate quarterly time series from 1960(1) to 1977(3) with variables

- **VST** value of stocks,
- **MB** monetary base,
- **RTPD** dollar rent on producer durables,
- **RTPS** dollar rent on producer structures,
- **XBC** production capacity for business output.

Source

The data was originally studied by Woglom (1981), the data set is given in Krämer and Sonnberger (1986).
References


Examples

data(valueofstocks)
lm(log(VST) ~., data=valueofstocks)

<table>
<thead>
<tr>
<th>wages</th>
<th>Wages</th>
</tr>
</thead>
</table>

Description

Wages Data.

Usage

data(wages)

Format

A multivariate yearly time series from 1960 to 1979 with variables

- **w** wages,
- **CPI** consumer price index,
- **u** unemployment,
- **mw** minimum wage.

Source

The data was originally studied by Nicols (1983), the data set is given in Krämer and Sonnberger (1986). Below we replicate a few examples from their book. Some of these results differ more or less seriously and are sometimes parameterized differently.

References


Examples

```r
data(wages)

## data transformation to include lagged series
mywages <- cbind(wages[,2], lag(wages[,2], k = -1), lag(wages[,2], k = -2))
colnames(mywages) <- c(colnames(wages), "CPI2", "CPI3")
mywages <- window(mywages, start=1962, end=1979)

## page 142, fit Nichols OLS model
## equation (6.10)
modelNichols <- w ~ CPI + CPI2 + CPI3 + u + mw
lm(modelNichols, data = mywages)

## page 143, fit test statistics in table 6.11
# *************************************
if(require(strucchange, quietly = TRUE)) {
## Chow 1972
sctest(modelNichols, point=c(1971,1), data=mywages, type="Chow")
}

## Breusch-Pagan
bptest(modelNichols, data=mywages, studentize=FALSE)
bptest(modelNichols, data=mywages)

## RESET (a)-(b)
reset(modelNichols, data=mywages)
reset(modelNichols, power=2, type="regressor", data=mywages)

## Harvey-Collier
harvtest(modelNichols, order.by = ~ CPI, data=mywages)
harvtest(modelNichols, order.by = ~ CPI2, data=mywages)
harvtest(modelNichols, order.by = ~ CPI3, data=mywages)
harvtest(modelNichols, order.by = ~ u, data=mywages)

## Rainbow
raintest(modelNichols, order.by = "mahalanobis", data=mywages)
```

---

### waldtest

**Wald Test of Nested Models**

**Description**

`waldtest` is a generic function for carrying out Wald tests. The default method can be employed for comparing nested (generalized) linear models (see details below).

**Usage**

```r
waldtest(object, ...)
```
## Default S3 method:
waldtest(object, ..., vcov = NULL,
        test = c("Chisq", "F"), name = NULL)

## S3 method for class 'formula'
waldtest(object, ..., data = list())

## S3 method for class 'lm'
waldtest(object, ..., test = c("F", "Chisq"))

### Arguments

- **object**: an object. See below for details.
- **...**: further object specifications passed to methods. See below for details.
- **vcov**: a function for estimating the covariance matrix of the regression coefficients, e.g., *vcovHC*. If only two models are compared it can also be the covariance matrix of the more general model.
- **test**: character specifying whether to compute the large sample Chi-squared statistic (with asymptotic Chi-squared distribution) or the finite sample F statistic (with approximate F distribution).
- **name**: a function for extracting a suitable name/description from a fitted model object. By default the name is queried by calling *formula*.
- **data**: a data frame containing the variables in the model.

### Details

waldtest is intended to be a generic function for comparisons of models via Wald tests. The default method consecutively compares the fitted model object *object* with the models passed in *...*. Instead of passing the fitted model objects in *...*, several other specifications are possible. For all objects in *list(object, ...)* the function tries to consecutively compute fitted models using the following updating algorithm:

1. For each two consecutive objects, *object1* and *object2* say, try to turn *object2* into a fitted model that can be compared to (the already fitted model object) *object1*.
2. If *object2* is numeric, the corresponding element of `attr(terms(object1), "term.labels")` is selected to be omitted.
3. If *object2* is a character, the corresponding terms are included into an update formula like `. ~ . - term2a - term2b`.
4. If *object2* is a formula, then compute the fitted model via `update(object1, object2)`.

Consequently, the models in *...* can be specified as integers, characters (both for terms that should be eliminated from the previous model), update formulas or fitted model objects. Except for the last case, the existence of an `update` method is assumed. See also the examples for an illustration.

Subsequently, a Wald test for each two consecutive models is carried out. This is similar to *anova* (which typically performs likelihood-ratio tests), but with a few differences. If only one fitted model object is specified, it is compared to the trivial model (with only an intercept). The test can
be either the finite sample F statistic or the asymptotic Chi-squared statistic ($F = \frac{\text{Chisq}}{k}$ if $k$ is the difference in degrees of freedom). The covariance matrix is always estimated on the more general of two subsequent models (and not only in the most general model overall). If vcov is specified, HC and HAC estimators can also be plugged into waldtest.

The default method is already very general and applicable to a broad range of fitted model objects, including lm and glm objects. It can be easily made applicable to other model classes as well by providing suitable methods to the standard generics terms (for determining the variables in the model along with their names), update (unless only fitted model objects are passed to waldtest, as mentioned above), nobs (or residuals, used for determining the number of observations), df.residual (needed only for the F statistic), coef (for extracting the coefficients; needs to be named matching the names in terms), vcov (can be user-supplied; needs to be named matching the names in terms). Furthermore, some means of determining a suitable name for a fitted model object can be specified (by default this is taken to be the result of a call to formula, if available).

The "formula" method fits a lm first and then calls the "lm" method. The "lm" method just calls the default method, but sets the default test to be the F test.

Value

An object of class "anova" which contains the residual degrees of freedom, the difference in degrees of freedom, Wald statistic (either "Chisq" or "F") and corresponding p value.

See Also

coefftest, anova, linearHypothesis

Examples

```r
## fit two competing, non-nested models and their encompassing
## model for aggregate consumption, as in Greene (1993),
## Examples 7.11 and 7.12

## load data and compute lags
data(USDistLag)
usdl <- na.contiguous(cbind(USDistLag, lag(USDistLag, k = -1)))
colnames(usdl) <- c("con", "gnp", "con1", "gnp1")

## C(t) = a0 + a1*Y(t) + a2*C(t-1) + u
fm1 <- lm(con ~ gnp + con1, data = usdl)

## C(t) = b0 + b1*Y(t) + b2*Y(t-1) + v
fm2 <- lm(con ~ gnp + gnp1, data = usdl)

## Encompassing model
fm3 <- lm(con ~ gnp + con1 + gnp1, data = usdl)

## a simple ANOVA for fm3 vs. fm2
waldtest(fm3, fm2)
anova(fm3, fm2)
```

```r
## as df = 1, the test is equivalent to the corresponding t test in
coefftest(fm3)
```
## various equivalent specifications of the two models
waldtest(fm3, fm2)
waldtest(fm3, 2)
waldtest(fm3, "con1")
waldtest(fm3, . - . - con1)

## comparing more than one model
## (equivalent to the encompassing test)
waldtest(fm1, fm3, fm2)
encomptest(fm1, fm2)

## using the asymptotic Chisq statistic
waldtest(fm3, fm2, test = "Chisq")
## plugging in a HC estimator
if(require(sandwich)) waldtest(fm3, fm2, vcov = vcovHC)
Index

* datasets
  bondyield, 4
  ChickEgg, 8
  currencysubstitution, 13
  ftemp, 18
  growthofmoney, 22
  jocci, 26
  Mandible, 31
  moneydemand, 31
  unemployment, 38
  USDistLag, 39
  valueofstocks, 40
  wages, 41
* htest
  bgtest, 2
  bptest, 6
  coeftest, 9
  coxtest, 11
  dwtest, 14
  encomptest, 16
  gqtest, 19
  grangertest, 21
  harvtest, 23
  hmctest, 25
  jtest, 28
  lrtest, 29
  petest, 33
  raintest, 35
  resettest, 36
  waldtest, 42

AIC, 10
anova, 43, 44

bgtest, 2
BIC, 10
bondyield, 4, 7, 15, 20, 24, 25, 35, 37
bptest, 6
breakpoints.formula, 9
ChickEgg, 8, 21
coeff, 44
coeff.coeftest(coeftest), 9
coeffci(coeftest), 9
coefftest, 3, 9, 28, 33, 34, 44
confint, 10
confint.coeftest(coeftest), 9
coxtest, 11, 17, 29, 34
currencysubstitution, 7, 13, 15, 20, 24, 25, 35, 37
df.residual, 44
df.residual.bgtest(bgtest), 2
df.residual.coeftest(coeftest), 9
dwtest, 3, 14
encomptest, 12, 16, 29, 34
formula, 30, 43, 44
ftemp, 18
fyff(jocci), 26
glm, 9, 44
gmdc(jocci), 26
gqtest, 19
grangertest, 21
growthofmoney, 7, 15, 20, 22, 24, 25, 35, 37
harvtest, 23
hmctest, 25
ip(jocci), 26
jocci, 26
jtest, 12, 17, 28, 34
lhur(jocci), 26
linearHypothesis, 44
lm, 7, 9, 10, 16, 20, 24, 26, 30, 36, 37, 44
logLik, 10, 30
logLik.coeftest(coeftest), 9
INDEX

lrtest, 29

Mandible, 31
moneydemand, 7, 15, 20, 24, 25, 31, 35, 37

ncvTest, 7
nobs, 10, 44
nobs.coeftest (coeftest), 9

petest, 33
print.coeftest (coeftest), 9
pw561 (jocci), 26

raintest, 35
reset (resettest), 36
resettest, 36
residuals, 44

sandwich, 9
survreg, 9

terms, 44

unemployment, 7, 15, 20, 24, 25, 35, 37, 38
update, 30, 43, 44
USDistLag, 39

valueofstocks, 40
vcov, 44
vcov.bgtest (bgtest), 2
vcovCL, 9
vcovHAC, 9
vcovHC, 9, 17, 28, 33, 43

wages, 7, 15, 20, 24, 25, 35, 37, 41
waldtest, 10, 17, 21, 30, 36, 42