Package 'tseries'

April 15, 2018

Version 0.10-44

Title Time Series Analysis and Computational Finance

Description Time series analysis and computational finance.

Depends R (>= 2.10.0)

Imports graphics, stats, utils, quadprog, zoo, quantmod (>= 0.4.9)

License GPL-2

NeedsCompilation yes

Author Adrian Trapletti [aut],
Kurt Hornik [aut, cre],
Blake LeBaron [ctb] (BDS test code)

Maintainer Kurt Hornik <Kurt.Hornik@R-project.org>

Repository CRAN

Date/Publication 2018-04-15 16:35:20 UTC

R topics documented:

`adf.test` *Augmented Dickey–Fuller Test*

Description

Computes the Augmented Dickey-Fuller test for the null that `x` has a unit root.

Usage

```
adf.test(x, alternative = c("stationary", "explosive"),
         k = trunc((length(x)-1)^(1/3)))
```

Arguments

`x`	a numeric vector or time series.
`alternative`	indicates the alternative hypothesis and must be one of `"stationary"` (default) or `"explosive"`. You can specify just the initial letter.
`k`	the lag order to calculate the test statistic.

Details

The general regression equation which incorporates a constant and a linear trend is used and the t-statistic for a first order autoregressive coefficient equals one is computed. The number of lags used in the regression is k. The default value of `trunc((length(x)-1)^(1/3))` corresponds to the suggested upper bound on the rate at which the number of lags, k, should be made to grow with the sample size for the general `ARMA(p,q)` setup. Note that for k equals zero the standard Dickey-Fuller test is computed. The p-values are interpolated from Table 4.2, p. 103 of Banerjee et al. (1993). If the computed statistic is outside the table of critical values, then a warning message is generated.

Missing values are not allowed.

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.

Author(s)

A. Trapletti

References

A. Banerjee, J. J. Dolado, J. W. Galbraith, and D. F. Hendry (1993): *Cointegration, Error Correction, and the Econometric Analysis of Non-Stationary Data*, Oxford University Press, Oxford.

S. E. Said and D. A. Dickey (1984): Testing for Unit Roots in Autoregressive-Moving Average Models of Unknown Order. *Biometrika* **71**, 599–607.

See Also

`pp.test`

Examples

```
x <- rnorm(1000)  # no unit-root
adf.test(x)

y <- diffinv(x)   # contains a unit-root
adf.test(y)
```

arma *Fit ARMA Models to Time Series*

Description

Fit an ARMA model to a univariate time series by conditional least squares. For exact maximum likelihood estimation see arima0.

Usage

```
arma(x, order = c(1, 1), lag = NULL, coef = NULL,
     include.intercept = TRUE, series = NULL, qr.tol = 1e-07, ...)
```

Arguments

x	a numeric vector or time series.
order	a two dimensional integer vector giving the orders of the model to fit. order[1] corresponds to the AR part and order[2] to the MA part.
lag	a list with components ar and ma. Each component is an integer vector, specifying the AR and MA lags that are included in the model. If both, order and lag, are given, only the specification from lag is used.
coef	If given this numeric vector is used as the initial estimate of the ARMA coefficients. The preliminary estimator suggested in Hannan and Rissanen (1982) is used for the default initialization.
include.intercept	Should the model contain an intercept?
series	name for the series. Defaults to deparse(substitute(x)).
qr.tol	the tol argument for qr when computing the asymptotic standard errors of coef.
...	additional arguments for optim when fitting the model.

Details

The following parametrization is used for the ARMA(p,q) model:

$$y[t] = a[0] + a[1]y[t-1] + \ldots + a[p]y[t-p] + b[1]e[t-1] + \ldots + b[q]e[t-q] + e[t],$$

where $a[0]$ is set to zero if no intercept is included. By using the argument lag, it is possible to fit a parsimonious submodel by setting arbitrary $a[i]$ and $b[i]$ to zero.

arma uses optim to minimize the conditional sum-of-squared errors. The gradient is computed, if it is needed, by a finite-difference approximation. Default initialization is done by fitting a pure high-order AR model (see ar.ols). The estimated residuals are then used for computing a least squares estimator of the full ARMA model. See Hannan and Rissanen (1982) for details.

Value

A list of class "arma" with the following elements:

lag	the lag specification of the fitted model.
coef	estimated ARMA coefficients for the fitted model.
css	the conditional sum-of-squared errors.
n.used	the number of observations of x.
residuals	the series of residuals.
fitted.values	the fitted series.
series	the name of the series x.
frequency	the frequency of the series x.
call	the call of the arma function.
vcov	estimate of the asymptotic-theory covariance matrix for the coefficient estimates.
convergence	The convergence integer code from optim.
include.intercept	Does the model contain an intercept?

Author(s)

A. Trapletti

References

E. J. Hannan and J. Rissanen (1982): Recursive Estimation of Mixed Autoregressive-Moving Average Order. *Biometrika* **69**, 81–94.

See Also

summary.arma for summarizing ARMA model fits; arma-methods for further methods; arima0, ar.

Examples

```
data(tcm)
r <- diff(tcm10y)
summary(r.arma <- arma(r, order = c(1, 0)))
summary(r.arma <- arma(r, order = c(2, 0)))
summary(r.arma <- arma(r, order = c(0, 1)))
summary(r.arma <- arma(r, order = c(0, 2)))
summary(r.arma <- arma(r, order = c(1, 1)))
plot(r.arma)

data(nino)
s <- nino3.4
summary(s.arma <- arma(s, order=c(20,0)))
summary(s.arma
```

```
        <- arma(s, lag=list(ar=c(1,3,7,10,12,13,16,17,19),ma=NULL)))
acf(residuals(s.arma), na.action=na.remove)
pacf(residuals(s.arma), na.action=na.remove)
summary(s.arma
        <- arma(s, lag=list(ar=c(1,3,7,10,12,13,16,17,19),ma=12)))
summary(s.arma
        <- arma(s, lag=list(ar=c(1,3,7,10,12,13,16,17),ma=12)))
plot(s.arma)
```

`arma-methods` *Methods for Fitted ARMA Models*

Description

Methods for fitted ARMA model objects.

Usage

```
## S3 method for class 'arma'
coef(object, ...)
## S3 method for class 'arma'
vcov(object, ...)
## S3 method for class 'arma'
residuals(object, ...)
## S3 method for class 'arma'
fitted(object, ...)
## S3 method for class 'arma'
print(x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'arma'
plot(x, ask = interactive(), ...)
```

Arguments

`object, x`	an object of class `"arma"`; usually, a result of a call to `arma`.
`digits`	see `printCoefmat`.
`ask`	Should the `plot` method work interactively? See `interactive`.
`...`	further arguments passed to or from other methods.

Value

For `coef`, a numeric vector; for `vcov`, a numeric matrix; for `residuals` and `fitted` a univariate time series; for `plot` and `print`, the fitted ARMA model object.

Author(s)

A. Trapletti

See Also

arma

---

bds.test *BDS Test*

Description

Computes and prints the BDS test statistic for the null that x is a series of i.i.d. random variables.

Usage

```
bds.test(x, m = 3, eps = seq(0.5 * sd(x), 2 * sd(x), length = 4),
         trace = FALSE)
```

Arguments

x	a numeric vector or time series.
m	an integer indicating that the BDS test statistic is computed for embedding dimensions 2, ..., m.
eps	a numeric vector of epsilon values for close points. The BDS test is computed for each element of eps. It should be set in terms of the standard deviation of x.
trace	a logical indicating whether some informational output is traced.

Details

This test examines the "spatial dependence" of the observed series. To do this, the series is embedded in m-space and the dependence of x is examined by counting "near" points. Points for which the distance is less than eps are called "near". The BDS test statistic is asymptotically standard Normal.

Missing values are not allowed.

There is a special print method for objects of class "bdstest" which by default uses 4 digits to format real numbers.

Value

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

statistic	the values of the test statistic.
p.value	the p-values of the test.
method	a character string indicating what type of test was performed.
parameter	a list with the components m and eps containing the embedding dimensions and epsilon values for which the statistic is computed.
data.name	a character string giving the name of the data.

Author(s)

B. LeBaron, Ported to R by A. Trapletti

References

J. B. Cromwell, W. C. Labys and M. Terraza (1994): *Univariate Tests for Time Series Models*, Sage, Thousand Oaks, CA, pages 32–36.

Examples

```
x <- rnorm(100)
bds.test(x)  # i.i.d. example

x <- c(rnorm(50), runif(50))
bds.test(x)  # not identically distributed

x <- quadmap(xi = 0.2, a = 4.0, n = 100)
bds.test(x)  # not independent
```

---

bev *Beveridge Wheat Price Index, 1500–1869.*

Description

Contains the well-known Beveridge Wheat Price Index which gives annual price data from 1500 to 1869, averaged over many locations in western and central Europe.

Usage

```
data(bev)
```

Format

A univariate time series with 370 observations. The object is of class `"ts"`.

Details

This data provides an example of long memory time series which has the appearance of being nonstationary in levels and yet also appears overdifferenced. See, e.g., Baillie (1996).

Source

Time Series Data Library: https://robjhyndman.com/TSDL/

References

R. T. Baillie (1996): Long Memory Processes and Fractional Integration in Econometrics. *Journal of Econometrics*, **73**, 5–59.

camp *Mount Campito Yearly Treering Data, -3435–1969.*

Description

Contains annual tree-ring measurements from Mount Campito from 3426 BC through 1969 AD.

Usage

```
data(camp)
```

Format

A univariate time series with 5405 observations. The object is of class `"ts"`.

Details

This series is a standard example for the concept of long memory time series.

The data was produced and assembled at the Tree Ring Laboratory at the University of Arizona, Tuscon.

Source

Time Series Data Library: https://robjhyndman.com/TSDL/

garch *Fit GARCH Models to Time Series*

Description

Fit a Generalized Autoregressive Conditional Heteroscedastic GARCH(p, q) time series model to the data by computing the maximum-likelihood estimates of the conditionally normal model.

Usage

```
garch(x, order = c(1, 1), series = NULL, control = garch.control(...), ...)

garch.control(maxiter = 200, trace = TRUE, start = NULL,
  grad = c("analytical","numerical"), abstol = max(1e-20, .Machine$double.eps^2),
  reltol = max(1e-10, .Machine$double.eps^(2/3)), xtol = sqrt(.Machine$double.eps),
   falsetol = 1e2 * .Machine$double.eps, ...)
```

Arguments

x	a numeric vector or time series.
order	a two dimensional integer vector giving the orders of the model to fit. order[2] corresponds to the ARCH part and order[1] to the GARCH part.
series	name for the series. Defaults to deparse(substitute(x)).
control	a list of control parameters as set up by garch.control.
maxiter	gives the maximum number of log-likelihood function evaluations maxiter and the maximum number of iterations 2*maxiter the optimizer is allowed to compute.
trace	logical. Trace optimizer output?
start	If given this numeric vector is used as the initial estimate of the GARCH coefficients. Default initialization is to set the GARCH parameters to slightly positive values and to initialize the intercept such that the unconditional variance of the initial GARCH is equal to the variance of x.
grad	character indicating whether analytical gradients or a numerical approximation is used for the optimization.
abstol	absolute function convergence tolerance.
reltol	relative function convergence tolerance.
xtol	coefficient-convergence tolerance.
falsetol	false convergence tolerance.
...	additional arguments for qr when computing the asymptotic covariance matrix.

Details

garch uses a Quasi-Newton optimizer to find the maximum likelihood estimates of the conditionally normal model. The first max(p, q) values are assumed to be fixed. The optimizer uses a hessian approximation computed from the BFGS update. Only a Cholesky factor of the Hessian approximation is stored. For more details see Dennis et al. (1981), Dennis and Mei (1979), Dennis and More (1977), and Goldfarb (1976). The gradient is either computed analytically or using a numerical approximation.

Value

A list of class "garch" with the following elements:

order	the order of the fitted model.
coef	estimated GARCH coefficients for the fitted model.
n.likeli	the negative log-likelihood function evaluated at the coefficient estimates (apart from some constant).
n.used	the number of observations of x.
residuals	the series of residuals.
fitted.values	the bivariate series of conditional standard deviation predictions for x.
series	the name of the series x.

`frequency`	the frequency of the series `x`.
`call`	the call of the `garch` function.
`vcov`	outer product of gradient estimate of the asymptotic-theory covariance matrix for the coefficient estimates.

Author(s)

A. Trapletti, the whole GARCH part; D. M. Gay, the FORTRAN optimizer

References

A. K. Bera and M. L. Higgins (1993): ARCH Models: Properties, Estimation and Testing. *J. Economic Surveys* **7** 305–362.

T. Bollerslev (1986): Generalized Autoregressive Conditional Heteroscedasticity. *Journal of Econometrics* **31**, 307–327.

R. F. Engle (1982): Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica* **50**, 987–1008.

J. E. Dennis, D. M. Gay, and R. E. Welsch (1981): Algorithm 573 — An Adaptive Nonlinear Least-Squares Algorithm. *ACM Transactions on Mathematical Software* **7**, 369–383.

J. E. Dennis and H. H. W. Mei (1979): Two New Unconstrained Optimization Algorithms which use Function and Gradient Values. *J. Optim. Theory Applic.* **28**, 453–482.

J. E. Dennis and J. J. More (1977): Quasi-Newton Methods, Motivation and Theory. *SIAM Rev.* **19**, 46–89.

D. Goldfarb (1976): Factorized Variable Metric Methods for Unconstrained Optimization. *Math. Comput.* **30**, 796–811.

See Also

`summary.garch` for summarizing GARCH model fits; `garch-methods` for further methods.

Examples

```
n <- 1100
a <- c(0.1, 0.5, 0.2)  # ARCH(2) coefficients
e <- rnorm(n)
x <- double(n)
x[1:2] <- rnorm(2, sd = sqrt(a[1]/(1.0-a[2]-a[3])))
for(i in 3:n)  # Generate ARCH(2) process
{
  x[i] <- e[i]*sqrt(a[1]+a[2]*x[i-1]^2+a[3]*x[i-2]^2)
}
x <- ts(x[101:1100])
x.arch <- garch(x, order = c(0,2))  # Fit ARCH(2)
summary(x.arch)                     # Diagnostic tests
plot(x.arch)

data(EuStockMarkets)
dax <- diff(log(EuStockMarkets))[,"DAX"]
```

```
dax.garch <- garch(dax)  # Fit a GARCH(1,1) to DAX returns
summary(dax.garch)       # ARCH effects are filtered. However,
plot(dax.garch)          # conditional normality seems to be violated
```

---

garch-methods *Methods for Fitted GARCH Models*

Description

Methods for fitted GARCH model objects.

Usage

```
## S3 method for class 'garch'
predict(object, newdata, genuine = FALSE, ...)
## S3 method for class 'garch'
coef(object, ...)
## S3 method for class 'garch'
vcov(object, ...)
## S3 method for class 'garch'
residuals(object, ...)
## S3 method for class 'garch'
fitted(object, ...)
## S3 method for class 'garch'
print(x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'garch'
plot(x, ask = interactive(), ...)
## S3 method for class 'garch'
logLik(object, ...)
```

Arguments

object, x	an object of class "garch"; usually, a result of a call to garch.
newdata	a numeric vector or time series to compute GARCH predictions. Defaults to eval(parse(text=object$series)).
genuine	a logical indicating whether a genuine prediction should be made, i.e., a prediction for which there is no target observation available.
digits	see printCoefmat.
ask	Should the plot method work interactively? See interactive.
...	further arguments passed to or from other methods.

Details

predict returns +/- the conditional standard deviation predictions from a fitted GARCH model.

coef returns the coefficient estimates.

vcov the associated covariance matrix estimate (outer product of gradients estimator).

residuals returns the GARCH residuals, i.e., the time series used to fit the model divided by the computed conditional standard deviation predictions for this series. Under the assumption of conditional normality the residual series should be i.i.d. standard normal.

fitted returns +/- the conditional standard deviation predictions for the series which has been used to fit the model.

plot graphically investigates normality and remaining ARCH effects for the residuals.

logLik returns the log-likelihood value of the GARCH(p, q) model represented by object evaluated at the estimated coefficients. It is assumed that first max(p, q) values are fixed.

Value

For predict a bivariate time series (two-column matrix) of predictions.

For coef, a numeric vector, for residuals and fitted a univariate (vector) and a bivariate time series (two-column matrix), respectively.

For plot and print, the fitted GARCH model object.

Author(s)

A. Trapletti

---

get.hist.quote *Download Historical Finance Data*

Description

Download historical financial data from a given data provider over the WWW.

Usage

```
get.hist.quote(instrument = "^gdax", start, end,
               quote = c("Open", "High", "Low", "Close"),
               provider = c("yahoo"), method = NULL,
               origin = "1899-12-30", compression = "d",
       retclass = c("zoo", "ts"), quiet = FALSE, drop = FALSE)
```

Arguments

instrument	a character string giving the name of the quote symbol to download. See the web page of the data provider for information about the available quote symbols.
start	an R object specifying the date of the start of the period to download. This must be in a form which is recognized by as.POSIXct, which includes R POSIX date/time objects, objects of class "date" (from package date) and "chron" and "dates" (from package chron), and character strings representing dates in ISO 8601 format. Defaults to 1992-01-02.
end	an R object specifying the end of the download period, see above. Defaults to yesterday.
quote	a character string or vector indicating whether to download opening, high, low, or closing quotes, or volume. For the default provider, this can be specified as "Open", "High", "Low", "Close", "Adjusted", and "Volume", respectively. Abbreviations are allowed.
provider	a character string with the name of the data provider. Currently, only "yahoo" is supported via getSymbols from package **quantmod** for the Yahoo Finance source. Provider "oanda" is no longer available.
method	No longer used.
origin	an R object specifying the origin of the Julian dates, see above. Defaults to 1899-12-30 (Popular spreadsheet programs internally also use Julian dates with this origin).
compression	Governs the granularity of the retrieved data; "d" for daily, "w" for weekly or "m" for monthly. Defaults to "d". For the provider "oanda", this argument is ignored.
retclass	character specifying which class the return value should have: can be either "zoo" (with "Date" index), or "ts" (with numeric index corresponding to days since origin).
quiet	logical. Should status messages (if any) be suppressed?
drop	logical. If TRUE the result is coerced to the lowest possible dimension. Default is FALSE.

Value

A time series containing the data either as a "zoo" series (default) or a "ts" series. The "zoo" series is created with zoo and has an index of class "Date". If a "ts" series is returned, the index is in physical time, i.e., weekends, holidays, and missing days are filled with NAs if not available. The time scale is given in Julian dates (days since the origin).

Author(s)

A. Trapletti

See Also

getSymbols for downloads from various providers; zoo, ts, as.Date, as.POSIXct,

Examples

```
con <- url("https://finance.yahoo.com")
if(!inherits(try(open(con), silent = TRUE), "try-error")) {
  close(con)
  x <- get.hist.quote(instrument = "^gspc", start = "1998-01-01",
                      quote = "Close")
  plot(x)

  x <- get.hist.quote(instrument = "ibm", quote = c("Cl", "Vol"))
  plot(x, main = "International Business Machines Corp")

  spc <- get.hist.quote(instrument = "^gspc", start = "1998-01-01",
         quote = "Close")
  ibm <- get.hist.quote(instrument = "ibm",  start = "1998-01-01",
         quote = "Adj")
  require("zoo") # For merge() method.
  x <- merge(spc, ibm)
  plot(x, main = "IBM vs S&P 500")
}
```

---

ice.river *Icelandic River Data*

Description

Contains the Icelandic river data as presented in Tong (1990), pages 432–440.

Usage

```
data(ice.river)
```

Format

4 univariate time series `flow.vat`, `flow.jok`, `prec`, and `temp`, each with 1095 observations and the joint series `ice.river`.

Details

The series are daily observations from Jan. 1, 1972 to Dec. 31, 1974 on 4 variables: `flow.vat`, mean daily flow of Vatnsdalsa river (cms), `flow.jok`, mean daily flow of Jokulsa Eystri river (cms), `prec`, daily precipitation in Hveravellir (mm), and mean daily temperature in Hveravellir (deg C).

These datasets were introduced into the literature in a paper by Tong, Thanoon, and Gudmundsson (1985).

Source

Time Series Data Library: https://robjhyndman.com/TSDL/

References

H. Tong (1990): *Non-Linear Time Series, A Dynamical System Approach*. Oxford University Press, Oxford.

H. Tong, B. Thanoon, and G. Gudmundsson (1985): Threshold time series modelling of two Icelandic riverflow systems. *Water Resources Bulletin*, **21**, 651–661.

`irts` *Irregularly Spaced Time-Series*

Description

The function `irts` is used to create irregular time-series objects.

`as.irts` coerces an object to an irregularly spaced time-series. `is.irts` tests whether an object is an irregularly spaced time series.

Usage

```
irts(time, value)
as.irts(object)
is.irts(object)
```

Arguments

`time`	a numeric vector or a vector of class `"POSIXct"` representing the time-stamps of the irregular time-series object. The elements of the numeric vector are construed as the (signed) number of seconds since the beginning of 1970, see `POSIXct`.
`value`	a numeric vector or matrix representing the values of the irregular time-series object.
`object`	an R object to be coerced to an irregular time-series object or an R object to be tested whether it is an irregular time-series object.

Details

The function `irts` is used to create irregular time-series objects. These are scalar or vector valued time series indexed by a time-stamp of class `"POSIXct"`. Unlike objects of class `"ts"`, they can be used to represent irregularly spaced time-series.

Value

A list of class `"irts"` with the following elements:

`time`	a vector of class `"POSIXct"`.
`value`	a numeric vector or matrix.

Author(s)

A. Trapletti

See Also

ts, POSIXct, irts-methods, irts-functions

Examples

```
n <- 10
t <- cumsum(rexp(n, rate = 0.1))
v <- rnorm(n)
x <- irts(t, v)
x

as.irts(cbind(t, v))

is.irts(x)

# Multivariate
u <- rnorm(n)
irts(t, cbind(u, v))
```

---

irts-functions *Basic Functions for Irregular Time-Series Objects*

Description

Basic functions related to irregular time-series objects.

Usage

```
daysecond(object, tz = "GMT")
approx.irts(object, time, ...)
is.businessday(object, tz = "GMT")
is.weekend(object, tz = "GMT")
read.irts(file, format = "%Y-%m-%d %H:%M:%S", tz = "GMT", ...)
weekday(object, tz = "GMT")
write.irts(object, file = "", append = FALSE, quote = FALSE,
           sep = " ", eol = "\n", na = "NA", dec = ".",
           row.names = FALSE, col.names = FALSE, qmethod = "escape",
           format = "%Y-%m-%d %H:%M:%S", tz = "GMT", usetz = FALSE,
           format.value = NULL, ...)
```

Arguments

object	an object of class "irts"; usually, a result of a call to irts.
format, tz, usetz	formatting related arguments, see format.POSIXct.
time	an object of class "POSIXct" specifying the times at which to interpolate the irregularly spaced time-series.
file, append, quote, sep, eol, na, dec, row.names, col.names, qmethod	reading and writing related arguments, see read.table and write.table.
format.value	a string which specifies the formatting of the values when writing an irregular time-series object to a file. format.value is passed unchanged as argument format to the function formatC.
...	further arguments passed to or from other methods: for approx.irts passed to approx; for read.irts passed to read.table; for write.irts passed to data.frame.

Details

daysecond and weekday return the number of seconds since midnight (the same day) and the weekday as a decimal number (0-6, Sunday is 0), respectively.

is.businessday and is.weekend test which entries of an irregular time-series object are recorded on business days and weekends, respectively.

approx.irts interpolates an irregularly spaced time-series at prespecified times.

read.irts is the function to read irregular time-series objects from a file.

write.irts is the function to write irregular time-series objects to a file.

Value

For daysecond and weekday a vector of decimal numbers representing the number of seconds and the weekday, respectively.

For is.businessday and is.weekend a vector of "logical" representing the test results for each time.

For approx.irts, read.irts and write.irts an object of class "irts".

Author(s)

A. Trapletti

See Also

irts, irts-methods

Examples

```
n <- 10
t <- cumsum(rexp(n, rate = 0.1))
v <- rnorm(n)
x <- irts(t, v)

daysecond(x)
weekday(x)
is.businessday(x)
is.weekend(x)
x

approx.irts(x, seq(ISOdatetime(1970, 1, 1, 0, 0, 0, tz = "GMT"),
            by = "10 secs", length = 7), rule = 2)

## Not run:
file <- tempfile()

# To write an irregular time-series object to a file one might use
write.irts(x, file = file)

# To read an irregular time-series object from a file one might use
read.irts(file = file)

unlink(file)

## End(Not run)
```

---

irts-methods *Methods for Irregular Time-Series Objects*

Description

Methods for irregular time-series objects.

Usage

```
## S3 method for class 'irts'
lines(x, type = "l", ...)
## S3 method for class 'irts'
plot(x, type = "l", plot.type = c("multiple", "single"),
     xlab = "Time", ylab = NULL, main = NULL, ylim = NULL,
     oma = c(6, 0, 5, 0), ...)
## S3 method for class 'irts'
points(x, type = "p", ...)
## S3 method for class 'irts'
print(x, format = "%Y-%m-%d %H:%M:%S", tz = "GMT",
      usetz = TRUE, format.value = NULL, ...)
## S3 method for class 'irts'
```

```
time(x, ...)
## S3 method for class 'irts'
value(x, ...)
## S3 method for class 'irts'
x[i, j, ...]
```

Arguments

x	an object of class "irts"; usually, a result of a call to irts.
type, plot.type, xlab, ylab, main, ylim, oma	graphical arguments, see plot, points, lines, par, and plot.ts.
format, tz, usetz	formatting related arguments, see format.POSIXct.
format.value	a string which specifies the formatting of the values when printing an irregular time-series object. format.value is passed unchanged as argument format to the function formatC.
i, j	indices specifying the parts to extract from an irregular time-series object.
...	further arguments passed to or from other methods: for lines passed to lines; for plot passed to plot, plot.default, and mtext; for points passed to points; for print passed to formatC; for time, value, and [.irts unused.

Details

plot is the method for plotting irregular time-series objects.

points and lines are the methods for drawing a sequence of points as given by an irregular time-series object and joining the corresponding points with line segments, respectively.

print is the method for printing irregular time-series objects.

time and value are the methods for extracting the sequence of times and the sequence of values of an irregular time-series object.

[.irts is the method for extracting parts of irregular time-series objects.

Value

For time an object of class "POSIXct" representing the sequence of times. For value a vector or matrix representing the sequence of values.

For [.irts an object of class "irts" representing the extracted part.

For plot, points, lines, and print the irregular time-series object.

Author(s)

A. Trapletti

See Also

irts, irts-functions

Examples

```
n <- 10
t <- cumsum(rexp(n, rate = 0.1))
v <- rnorm(n)
x <- irts(t, v)

x
time(x)
value(x)
plot(x)
points(x)

t <- cumsum(c(t[1], rexp(n-1, rate = 0.2)))
v <- rnorm(n, sd = 0.1)
x <- irts(t, v)

lines(x, col = "red")
points(x, col = "red")

# Multivariate
t <- cumsum(rexp(n, rate = 0.1))
u <- rnorm(n)
v <- rnorm(n)
x <- irts(t, cbind(u, v))

x
x[,1]
x[1:3,]
x[1:3,1]
plot(x)
```

---

jarque.bera.test *Jarque–Bera Test*

Description

Tests the null of normality for x using the Jarque-Bera test statistic.

Usage

```
jarque.bera.test(x)
```

Arguments

x	a numeric vector or time series.

Details

This test is a joint statistic using skewness and kurtosis coefficients.

Missing values are not allowed.

Value

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

statistic	the value of the test statistic.
parameter	the degrees of freedom.
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.

Author(s)

A. Trapletti

References

J. B. Cromwell, W. C. Labys and M. Terraza (1994): *Univariate Tests for Time Series Models*, Sage, Thousand Oaks, CA, pages 20–22.

Examples

```
x <- rnorm(100)  # null
jarque.bera.test(x)

x <- runif(100)  # alternative
jarque.bera.test(x)
```

---

kpss.test *KPSS Test for Stationarity*

Description

Computes the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test for the null hypothesis that x is level or trend stationary.

Usage

```
kpss.test(x, null = c("Level", "Trend"), lshort = TRUE)
```

Arguments

x	a numeric vector or univariate time series.
null	indicates the null hypothesis and must be one of "Level" (default) or "Trend". You can specify just the initial letter.
lshort	a logical indicating whether the short or long version of the truncation lag parameter is used.

Details

To estimate `sigma^2` the Newey-West estimator is used. If `lshort` is `TRUE`, then the truncation lag parameter is set to `trunc(3*sqrt(n)/13)`, otherwise `trunc(10*sqrt(n)/14)` is used. The p-values are interpolated from Table 1 of Kwiatkowski et al. (1992). If the computed statistic is outside the table of critical values, then a warning message is generated.

Missing values are not handled.

Value

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

`statistic`	the value of the test statistic.
`parameter`	the truncation lag parameter.
`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.

Author(s)

A. Trapletti

References

D. Kwiatkowski, P. C. B. Phillips, P. Schmidt, and Y. Shin (1992): Testing the Null Hypothesis of Stationarity against the Alternative of a Unit Root. *Journal of Econometrics* **54**, 159–178.

See Also

`pp.test`

Examples

```
x <- rnorm(1000)  # is level stationary
kpss.test(x)

y <- cumsum(x)  # has unit root
kpss.test(y)

x <- 0.3*(1:1000)+rnorm(1000)  # is trend stationary
kpss.test(x, null = "Trend")
```

maxdrawdown *Maximum Drawdown or Maximum Loss*

Description

This function computes the maximum drawdown or maximum loss of the univariate time series (or vector) x.

Usage

```
maxdrawdown(x)
```

Arguments

x	a numeric vector or univariate time series.

Details

The max drawdown or max loss statistic is defined as the maximum value drop after one of the peaks of x. For financial instruments the max drawdown represents the worst investment loss for a buy-and-hold strategy invested in x.

Value

A list containing the following three components:

maxdrawdown	double representing the max drawdown or max loss statistic.
from	the index (or vector of indices) where the max drawdown period starts.
to	the index (or vector of indices) where the max drawdown period ends.

Author(s)

A. Trapletti

See Also

sterling

Examples

```
# Toy example
x <- c(1:10, 9:7, 8:14, 13:8, 9:20)
mdd <- maxdrawdown(x)
mdd

plot(x)
segments(mdd$from, x[mdd$from], mdd$to, x[mdd$from], col="grey")
segments(mdd$from, x[mdd$to], mdd$to, x[mdd$to], col="grey")
mid <- (mdd$from + mdd$to)/2
```

```
arrows(mid, x[mdd$from], mid, x[mdd$to], col="red", length = 0.16)

# Realistic example
data(EuStockMarkets)
dax <- log(EuStockMarkets[,"DAX"])
mdd <- maxdrawdown(dax)
mdd

plot(dax)
segments(time(dax)[mdd$from], dax[mdd$from],
         time(dax)[mdd$to], dax[mdd$from], col="grey")
segments(time(dax)[mdd$from], dax[mdd$to],
         time(dax)[mdd$to], dax[mdd$to], col="grey")
mid <- time(dax)[(mdd$from + mdd$to)/2]
arrows(mid, dax[mdd$from], mid, dax[mdd$to], col="red", length = 0.16)
```

`na.remove` *NA Handling Routines for Time Series*

Description

Observations with missing values in some of the variables are removed.

Usage

```
na.remove(object, ...)
```

Arguments

`object`	a numeric matrix, vector, univariate, or multivariate time series.
`...`	further arguments to be passed to or from methods.

Details

For `na.remove.ts` this changes the "intrinsic" time scale. It is assumed that both, the new and the old time scale are synchronized at the first and the last valid observation. In between, the new series is equally spaced in the new time scale.

Value

An object without missing values. The attribute `"na.removed"` contains the indices of the removed missing values in `object`.

Author(s)

A. Trapletti

See Also

na.omit, na.fail

Examples

```
x<-ts(c(5453.08,5409.24,5315.57,5270.53, # one and a half week stock index
        5211.66,NA,NA,5160.80,5172.37))  # data including a weekend
na.remove(x)  # eliminate weekend and introduce ``business'' time scale
```

NelPlo *Nelson–Plosser Macroeconomic Time Series*

Description

These are the extended Nelson-Plosser Data.

Usage

```
data(NelPlo)
```

Format

14 macroeconomic time series: cpi, ip, gnp.nom, vel, emp, int.rate, nom.wages, gnp.def, money.stock, gnp.real, stock.prices, gnp.capita, real.wages, and unemp and the joint series NelPlo.

Details

The series are of various lengths but all end in 1988. The data set contains the following series: consumer price index, industrial production, nominal GNP, velocity, employment, interest rate, nominal wages, GNP deflator, money stock, real GNP, stock prices (S&P500), GNP per capita, real wages, unemployment.

Source

Journal of Business and Economic Statistics data archive http://www.amstat.org/publications/jbes/

References

G. Koop and M. F. J. Steel (1994): A Decision-Theoretic Analysis of the Unit-Root Hypothesis using Mixtures of Elliptical Models. *Journal of Business and Economic Statistics*, **12**, 95–107.

nino *Sea Surface Temperature (SST) Nino 3 and Nino 3.4 Indices*

Description

These data constitutes of Nino Region 3 and Nino Region 3.4 SST indices.

Usage

```
data(nino)
```

Format

Two univariate time series `nino3` and `nino3.4` with 598 observations and the joint series `nino`.

Details

The measurements are given in degrees Celsius. The Nino 3 Region is bounded by 90W-150W and 5S-5N. The Nino 3.4 Region is bounded by 120W-170W and 5S-5N.

Source

Climate Prediction Center: http://www.cpc.ncep.noaa.gov/data/indices/

plotOHLC *Plot Open-High-Low-Close Bar Chart*

Description

Plots open-high-low-close bar chart of a (financial) time series.

Usage

```
plotOHLC(x, xlim = NULL, ylim = NULL, xlab = "Time", ylab, col = par("col"),
         bg = par("bg"), axes = TRUE, frame.plot = axes, ann = par("ann"),
         main = NULL, date = c("calendar", "julian"), format = "%Y-%m-%d",
         origin = "1899-12-30", ...)
```

Arguments

x	a multivariate time series object of class "mts".
xlim, ylim, xlab, ylab, col, bg, axes, frame.plot, ann, main	graphical arguments, see plot, plot.default and par.
date	a string indicating the type of x axis annotation. Default is calendar dates.
format	a string indicating the format of the x axis annotation if date == "calendar". For details see format.POSIXct.
origin	an R object specifying the origin of the Julian dates if date == "calendar". Defaults to 1899-12-30 (Popular spreadsheet programs internally also use Julian dates with this origin).
...	further graphical arguments passed to plot.window, title, axis, and box.

Details

Within an open-high-low-close bar chart, each bar represents price information for the time interval between the open and the close price. The left tick for each bar indicates the open price for the time interval. The right tick indicates the closing price for the time interval. The vertical length of the bar represents the price range for the time interval.

The time scale of x must be in Julian dates (days since the origin).

Author(s)

A. Trapletti

See Also

plot.default, format.POSIXct, get.hist.quote

Examples

```
con <- url("https://finance.yahoo.com")
if(!inherits(try(open(con), silent = TRUE), "try-error")) {
  close(con)
  ## Plot OHLC bar chart for the last 'nDays' days of the instrument
  ## 'instrument'

  nDays <- 50
  instrument <- "^gspc"

  start <- strftime(as.POSIXlt(Sys.time() - nDays * 24 * 3600),
                    format="%Y-%m-%d")
  end <- strftime(as.POSIXlt(Sys.time()), format = "%Y-%m-%d")
  x <- get.hist.quote(instrument = instrument, start = start, end = end,
                      retclass = "ts")

  plotOHLC(x, ylab = "price", main = instrument)
}
```

po.test *Phillips–Ouliaris Cointegration Test*

Description

Computes the Phillips-Ouliaris test for the null hypothesis that x is not cointegrated.

Usage

```
po.test(x, demean = TRUE, lshort = TRUE)
```

Arguments

x	a matrix or multivariate time series.
demean	a logical indicating whether an intercept is included in the cointegration regression or not.
lshort	a logical indicating whether the short or long version of the truncation lag parameter is used.

Details

The Phillips-Perron Z(alpha) statistic for a unit root in the residuals of the cointegration regression is computed, see also pp.test. The unit root is estimated from a regression of the first variable (column) of x on the remaining variables of x without a constant and a linear trend. To estimate sigma^2 the Newey-West estimator is used. If lshort is TRUE, then the truncation lag parameter is set to trunc(n/100), otherwise trunc(n/30) is used. The p-values are interpolated from Table Ia and Ib, p. 189 of Phillips and Ouliaris (1990). If the computed statistic is outside the table of critical values, then a warning message is generated.

The dimension of x is restricted to six variables. Missing values are not handled.

Value

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

statistic	the value of the test statistic.
parameter	the truncation lag parameter.
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.

Author(s)

A. Trapletti

References

P. C. B. Phillips and S. Ouliaris (1990): Asymptotic Properties of Residual Based Tests for Cointegration. *Econometrica* **58**, 165–193.

See Also

pp.test

Examples

```
x <- ts(diffinv(matrix(rnorm(2000),1000,2)))  # no cointegration
po.test(x)

x <- diffinv(rnorm(1000))
y <- 2.0-3.0*x+rnorm(x,sd=5)
z <- ts(cbind(x,y))  # cointegrated
po.test(z)
```

---

portfolio.optim *Portfolio Optimization*

Description

Computes an efficient portfolio from the given return series x in the mean-variance sense.

Usage

```
## Default S3 method:
portfolio.optim(x, pm = mean(x), riskless = FALSE,
                shorts = FALSE, rf = 0.0, reslow = NULL, reshigh = NULL,
                covmat = cov(x), ...)
```

Arguments

x	a numeric matrix or multivariate time series consisting of a series of returns.
pm	the desired mean portfolio return.
riskless	a logical indicating whether there is a riskless lending and borrowing rate.
shorts	a logical indicating whether shortsales on the risky securities are allowed.
rf	the riskfree interest rate.
reslow	a vector specifying the (optional) lower bound on allowed portfolio weights.
reshigh	a vector specifying the (optional) upper bound on allowed portfolio weights.
covmat	the covariance matrix of asset returns.
...	further arguments to be passed from or to methods.

Details

The computed portfolio has the desired expected return `pm` and no other portfolio exists, which has the same mean return, but a smaller variance. Inequality restrictions of the form $w_l \leq w \leq w_h$ can be imposed using the `reslow` and `reshigh` vectors. An alternative covariance matrix estimate can be supplied via the `covmat` argument. To solve the quadratic program, `solve.QP` is used.

`portfolio.optim` is a generic function with methods for multivariate `"ts"` and `default` for matrix.

Missing values are not allowed.

Value

A list containing the following components:

`pw`	the portfolio weights.
`px`	the returns of the overall portfolio.
`pm`	the expected portfolio return.
`ps`	the standard deviation of the portfolio returns.

Author(s)

A. Trapletti

References

E. J. Elton and M. J. Gruber (1991): *Modern Portfolio Theory and Investment Analysis*, 4th Edition, Wiley, NY, pp. 65-93.

C. Huang and R. H. Litzenberger (1988): *Foundations for Financial Economics*, Elsevier, NY, pp. 59-82.

See Also

`solve.QP`

Examples

```
x <- rnorm(1000)
dim(x) <- c(500,2)
res <- portfolio.optim(x)
res$pw

require("zoo") # For diff() method.
X <- diff(log(as.zoo(EuStockMarkets)))
res <- portfolio.optim(X)                ## Long only
res$pw
res <- portfolio.optim(X, shorts=TRUE)   ## Long/Short
res$pw
```

`pp.test` *Phillips–Perron Unit Root Test*

Description

Computes the Phillips-Perron test for the null hypothesis that `x` has a unit root.

Usage

```
pp.test(x, alternative = c("stationary", "explosive"),
        type = c("Z(alpha)", "Z(t_alpha)"), lshort = TRUE)
```

Arguments

`x`	a numeric vector or univariate time series.
`alternative`	indicates the alternative hypothesis and must be one of `"stationary"` (default) or `"explosive"`. You can specify just the initial letter.
`type`	indicates which variant of the test is computed and must be one of `"Z(alpha)"` (default) or `"Z(t_alpha)"`.
`lshort`	a logical indicating whether the short or long version of the truncation lag parameter is used.

Details

The general regression equation which incorporates a constant and a linear trend is used and the `Z(alpha)` or `Z(t_alpha)` statistic for a first order autoregressive coefficient equals one are computed. To estimate `sigma^2` the Newey-West estimator is used. If `lshort` is `TRUE`, then the truncation lag parameter is set to `trunc(4*(n/100)^0.25)`, otherwise `trunc(12*(n/100)^0.25)` is used. The p-values are interpolated from Table 4.1 and 4.2, p. 103 of Banerjee et al. (1993). If the computed statistic is outside the table of critical values, then a warning message is generated.

Missing values are not handled.

Value

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

`statistic`	the value of the test statistic.
`parameter`	the truncation lag parameter.
`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.

Author(s)

A. Trapletti

References

A. Banerjee, J. J. Dolado, J. W. Galbraith, and D. F. Hendry (1993): *Cointegration, Error Correction, and the Econometric Analysis of Non-Stationary Data*, Oxford University Press, Oxford.

P. Perron (1988): Trends and Random Walks in Macroeconomic Time Series. *Journal of Economic Dynamics and Control* **12**, 297–332.

See Also

adf.test

Examples

```
x <- rnorm(1000)  # no unit-root
pp.test(x)

y <- cumsum(x)  # has unit root
pp.test(y)
```

---

quadmap *Quadratic Map (Logistic Equation)*

Description

Computes the quadratic map simulation.

Usage

```
quadmap(xi = 0.2, a = 4.0, n = 1000)
```

Arguments

xi	the initial value for the iteration.
a	the quadratic map parameter.
n	the length of the simulated series.

Value

A vector containing the simulated series.

Author(s)

A. Trapletti

Examples

```
x <- quadmap()
acf(x, 10)
```

`read.matrix` *Read Matrix Data*

Description

Reads a matrix data file.

Usage

```
read.matrix(file, header = FALSE, sep = "", skip = 0)
```

Arguments

`file`	the name of the file which the data are to be read from.
`header`	a logical value indicating whether the file contains the names of the columns as its first line.
`sep`	the field separator character. Values on each line of the file are separated by this character.
`skip`	the number of lines of the data file to skip before beginning to read data.

Details

Usually each row of the file represents an observation and each column contains a variable. The first row possibly contains the names of the variables (columns).

`read.matrix` might be more efficient than `read.table` for very large data sets.

Author(s)

A. Trapletti

See Also

`read.table`.

Examples

```
x <- matrix(0, 10, 10)
write(x, "test", ncolumns=10)
x <- read.matrix("test")
x
unlink("test")
```

read.ts *Read Time Series Data*

Description

Reads a time series file.

Usage

```
read.ts(file, header = FALSE, sep = "", skip = 0, ...)
```

Arguments

file	the name of the file which the data are to be read from. Each line of the file contains one observation of the variables.
header	a logical value indicating whether the file contains the names of the variables as its first line.
sep	the field separator character. Values on each line of the file are separated by this character.
skip	the number of lines of the data file to skip before beginning to read data.
...	Additional arguments for ts such as, e.g., start.

Details

Each row of the file represents an observation and each column contains a variable. The first row possibly contains the names of the variables.

Author(s)

A. Trapletti

See Also

ts.

Examples

```
data(sunspots)
st <- start(sunspots)
fr <- frequency(sunspots)
write(sunspots, "sunspots", ncolumns=1)
x <- read.ts("sunspots", start=st, frequency=fr)
plot(x)
unlink("sunspots")
```

`runs.test` *Runs Test*

Description

Computes the runs test for randomness of the dichotomous (binary) data series `x`.

Usage

```
runs.test(x, alternative = c("two.sided", "less", "greater"))
```

Arguments

`x`	a dichotomous factor.
`alternative`	indicates the alternative hypothesis and must be one of `"two.sided"` (default), `"less"`, or `"greater"`. You can specify just the initial letter.

Details

This test searches for randomness in the observed data series `x` by examining the frequency of runs. A "run" is defined as a series of similar responses.

Note, that by using the alternative `"less"` the null of randomness is tested against some kind of "under-mixing" ("trend"). By using the alternative `"greater"` the null of randomness is tested against some kind of "over-mixing" ("mean-reversion").

Missing values are not allowed.

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.
`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.

Author(s)

A. Trapletti

References

S. Siegel (1956): *Nonparametric Statistics for the Behavioural Sciences*, McGraw-Hill, New York.

S. Siegel and N. J. Castellan (1988): *Nonparametric Statistics for the Behavioural Sciences*, 2nd edn, McGraw-Hill, New York.

Examples

```
x <- factor(sign(rnorm(100)))  # randomness
runs.test(x)

x <- factor(rep(c(-1,1),50))  # over-mixing
runs.test(x)
```

`seqplot.ts` *Plot Two Time Series*

Description

Plot two time series on the same plot frame.

Usage

```
seqplot.ts(x, y, colx = "black", coly = "red", typex = "l",
           typey = "l", pchx = 1, pchy = 1, ltyx = "solid",
           ltyy = "solid", oma = c(6, 0, 5, 0), ann = par("ann"),
           xlab = "Time", ylab = deparse(substitute(x)), main = NULL)
```

Arguments

`x, y`	the time series.
`colx, coly`	color code or name for the x and y series, see `colors`, `palette`.
`typex, typey`	what type of plot should be drawn for the x and y series, see `plot`.
`pchx, pchy`	character or integer code for kind of points/lines for the x and y series.
`ltyx, ltyy`	line type code for the x and y series, see `lines`.
`oma`	a vector giving the size of the outer margins in lines of text, see `par`.
`ann`	annotate the plots? See `par`.
`xlab, ylab`	titles for the x and y axis.
`main`	an overall title for the plot.

Details

Unlike `plot.ts` the series can have different time bases, but they should have the same frequency. Unlike `ts.plot` the series can be plotted in different styles and for multivariate x and y the common variables are plotted together in a separate array element.

Value

None.

Author(s)

A. Trapletti

See Also

ts, plot.ts

Examples

```
data(USeconomic)
x <- ts.union(log(M1), log(GNP), rs, rl)
m.ar <- ar(x, method = "ols", order.max = 5)
y <- predict(m.ar, x, n.ahead = 200, se.fit = FALSE)
seqplot.ts(x, y)
```

---

sharpe *Sharpe Ratio*

Description

This function computes the Sharpe ratio of the univariate time series (or vector) x.

Usage

```
sharpe(x, r = 0, scale = sqrt(250))
```

Arguments

x	a numeric vector or univariate time series corresponding to a portfolio's cumulated returns.
r	the risk free rate. Default corresponds to using portfolio returns not in excess of the riskless return.
scale	a scale factor. Default corresponds to an annualization when working with daily financial time series data.

Details

The Sharpe ratio is defined as a portfolio's mean return in excess of the riskless return divided by the portfolio's standard deviation. In finance the Sharpe Ratio represents a measure of the portfolio's risk-adjusted (excess) return.

Value

a double representing the Sharpe ratio.

Author(s)

A. Trapletti

See Also

sterling

Examples

```
data(EuStockMarkets)
dax <- log(EuStockMarkets[,"DAX"])
ftse <- log(EuStockMarkets[,"FTSE"])
sharpe(dax)
sharpe(ftse)
```

---

sterling *Sterling Ratio*

Description

This function computes the Sterling ratio of the univariate time series (or vector) x.

Usage

```
sterling(x)
```

Arguments

x	a numeric vector or univariate time series corresponding to a portfolio's cumulated returns.

Details

The Sterling ratio is defined as a portfolio's overall return divided by the portfolio's maxdrawdown statistic. In finance the Sterling Ratio represents a measure of the portfolio's risk-adjusted return.

Value

a double representing the Sterling ratio.

Author(s)

A. Trapletti

See Also

maxdrawdown, sharpe

Examples

```
data(EuStockMarkets)
dax <- log(EuStockMarkets[,"DAX"])
ftse <- log(EuStockMarkets[,"FTSE"])
sterling(dax)
sterling(ftse)
```

summary.arma *Summarizing ARMA Model Fits*

Description

Methods for creating and printing summaries of ARMA model fits.

Usage

```
## S3 method for class 'arma'
summary(object, ...)
## S3 method for class 'summary.arma'
print(x, digits = max(3, getOption("digits") - 3),
      signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	an object of class "arma"; usually, a result of a call to arma.
x	an object of class "summary.arma"; usually, a result of a call to the summary method for objects of class "arma".
digits, signif.stars	see printCoefmat.
...	further arguments passed to or from other methods.

Details

The summary method computes the asymptotic standard errors of the coefficient estimates from the numerically differentiated Hessian matrix approximation. The AIC is computed from the conditional sum-of-squared errors and not from the true maximum likelihood function. That may be problematic.

Value

A list of class "summary.arma".

See Also

arma

summary.garch *Summarizing GARCH Model Fits*

Description

Methods for creating and printing summaries of GARCH model fits.

Usage

```
## S3 method for class 'garch'
summary(object, ...)
## S3 method for class 'summary.garch'
print(x, digits = max(3, getOption("digits") - 3),
      signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	an object of class "garch"; usually, a result of a call to garch.
x	an object of class "summary.garch"; usually, a result of a call to the summary method for objects of class "garch".
digits, signif.stars	see printCoefmat.
...	further arguments passed to or from other methods.

Details

summary computes the asymptotic standard errors of the coefficient estimates from an outer-product approximation of the Hessian evaluated at the estimates, see Bollerslev (1986). It furthermore tests the residuals for normality and remaining ARCH effects, see jarque.bera.test and Box.test.

Value

A list of class "summary.garch".

References

T. Bollerslev (1986): Generalized Autoregressive Conditional Heteroscedasticity. *Journal of Econometrics* **31**, 307–327.

See Also

garch

surrogate *Generate Surrogate Data and Statistics*

Description

Generates ns surrogate samples from the original data x and computes the standard error and the bias of statistic as in a bootstrap setup, if statistic is given.

Usage

```
surrogate(x, ns = 1, fft = FALSE, amplitude = FALSE,
          statistic = NULL, ...)
```

Arguments

x	a numeric vector or time series.
ns	the number of surrogate series to compute.
fft	a logical indicating whether phase randomized surrogate data is generated.
amplitude	a logical indicating whether amplitude-adjusted surrogate data is computed.
statistic	a function which when applied to a time series returns a vector containing the statistic(s) of interest.
...	Additional arguments for statistic which are passed unchanged each time it is called.

Details

If fft is FALSE, then x is mixed in temporal order, so that all temporal dependencies are eliminated, but the histogram of the original data is preserved. If fft is TRUE, then surrogate data with the same spectrum as x is computed by randomizing the phases of the Fourier coefficients of x. If in addition amplitude is TRUE, then also the amplitude distribution of the original series is preserved.

Note, that the interpretation of the computed standard error and bias is different than in a bootstrap setup.

To compute the phase randomized surrogate and the amplitude adjusted data algorithm 1 and 2 from Theiler et al. (1992), pp. 183, 184 are used.

Missing values are not allowed.

Value

If statistic is NULL, then it returns a matrix or time series with ns columns and length(x) rows containing the surrogate data. Each column contains one surrogate sample.

If statistic is given, then a list of class "resample.statistic" with the following elements is returned:

statistic	the results of applying statistic to each of the simulated time series.

orig.statistic	the results of applying statistic to the original series.
bias	the bias of the statistics computed as in a bootstrap setup.
se	the standard error of the statistics computed as in a bootstrap setup.
call	the original call of surrogate.

Author(s)

A. Trapletti

References

J. Theiler, B. Galdrikian, A. Longtin, S. Eubank, and J. D. Farmer (1992): Using Surrogate Data to Detect Nonlinearity in Time Series, in *Nonlinear Modelling and Forecasting*, Eds. M. Casdagli and S. Eubank, Santa Fe Institute, Addison Wesley, 163–188.

See Also

sample, tsbootstrap

Examples

```
x <- 1:10  # Simple example
surrogate(x)

n <- 500  # Generate AR(1) process
e <- rnorm(n)
x <- double(n)
x[1] <- rnorm(1)
for(i in 2:n) {
  x[i] <- 0.4 * x[i-1] + e[i]
}
x <- ts(x)

theta <- function(x)  # Autocorrelations up to lag 10
  return(acf(x, plot=FALSE)$acf[2:11])

surrogate(x, ns=50, fft=TRUE, statistic=theta)
```

---

tcm *Monthly Yields on Treasury Securities*

Description

This data set contains monthly 1 year, 3 year, 5 year, and 10 year yields on treasury securities at constant, fixed maturity.

Usage

```
data(tcm)
```

Format

4 univariate time series `tcm1y`, `tcm3y`, `tcm5y`, and `tcm10y` and the joint series `tcm`.

Details

The yields at constant fixed maturity have been constructed by the Treasury Department, based on the most actively traded marketable treasury securities.

Source

U.S. Fed http://www.federalreserve.gov/Releases/H15/data.htm

---

tcmd *Daily Yields on Treasury Securities*

Description

This data set contains daily 1 year, 3 year, 5 year, and 10 year yields on treasury securities at constant, fixed maturity.

Usage

```
data(tcmd)
```

Format

4 univariate time series `tcm1yd`, `tcm3yd`, `tcm5yd`, and `tcm10yd` and the joint series `tcmd`.

Details

The yields at constant fixed maturity have been constructed by the Treasury Department, based on the most actively traded marketable treasury securities.

Daily refers to business days, i.e., weekends and holidays are eliminated.

Source

U.S. Fed http://www.federalreserve.gov/Releases/H15/data.htm

`terasvirta.test` *Teraesvirta Neural Network Test for Nonlinearity*

Description

Generically computes Teraesvirta's neural network test for neglected nonlinearity either for the time series `x` or the regression `y~x`.

Usage

```
## S3 method for class 'ts'
terasvirta.test(x, lag = 1, type = c("Chisq","F"),
                scale = TRUE, ...)
## Default S3 method:
terasvirta.test(x, y, type = c("Chisq","F"),
                scale = TRUE, ...)
```

Arguments

`x`	a numeric vector, matrix, or time series.
`y`	a numeric vector.
`lag`	an integer which specifies the model order in terms of lags.
`type`	a string indicating whether the Chi-Squared test or the F-test is computed. Valid types are `"Chisq"` and `"F"`.
`scale`	a logical indicating whether the data should be scaled before computing the test statistic. The default arguments to `scale` are used.
`...`	further arguments to be passed from or to methods.

Details

The null is the hypotheses of linearity in "mean". This test uses a Taylor series expansion of the activation function to arrive at a suitable test statistic. If `type` equals `"F"`, then the F-statistic instead of the Chi-Squared statistic is used in analogy to the classical linear regression.

Missing values are not allowed.

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.
`method`	a character string indicating what type of test was performed.
`parameter`	a list containing the additional parameters used to compute the test statistic.
`data.name`	a character string giving the name of the data.
`arguments`	additional arguments used to compute the test statistic.

Author(s)

A. Trapletti

References

T. Teraesvirta, C. F. Lin, and C. W. J. Granger (1993): Power of the Neural Network Linearity Test. *Journal of Time Series Analysis* 14, 209-220.

See Also

white.test

Examples

```
n <- 1000

x <- runif(1000, -1, 1)  # Non-linear in ``mean'' regression
y <- x^2 - x^3 + 0.1*rnorm(x)
terasvirta.test(x, y)

## Is the polynomial of order 2 misspecified?
terasvirta.test(cbind(x,x^2,x^3), y)

## Generate time series which is nonlinear in ``mean''
x[1] <- 0.0
for(i in (2:n)) {
  x[i] <- 0.4*x[i-1] + tanh(x[i-1]) + rnorm(1, sd=0.5)
}
x <- as.ts(x)
plot(x)
terasvirta.test(x)
```

---

tsbootstrap *Bootstrap for General Stationary Data*

Description

tsbootstrap generates bootstrap samples for general stationary data and computes the bootstrap estimate of standard error and bias if a statistic is given.

Usage

```
tsbootstrap(x, nb = 1, statistic = NULL, m = 1, b = NULL,
            type = c("stationary","block"), ...)
```

Arguments

x	a numeric vector or time series giving the original data.
nb	the number of bootstrap series to compute.
statistic	a function which when applied to a time series returns a vector containing the statistic(s) of interest.
m	the length of the basic blocks in the block of blocks bootstrap.
b	if type is "stationary", then b is the mean block length. If type is "block", then b is the fixed block length.
type	the type of bootstrap to generate the simulated time series. The possible input values are "stationary" (stationary bootstrap with mean block length b) and "block" (blockwise bootstrap with block length b). Default to "stationary".
...	additional arguments for statistic which are passed unchanged each time statistic is called.

Details

If type is "stationary", then the stationary bootstrap scheme with mean block length b according to Politis and Romano (1994) is computed. For type equals "block", the blockwise bootstrap with block length b according to Kuensch (1989) is used.

If m > 1, then the block of blocks bootstrap is computed (see Kuensch, 1989). The basic sampling scheme is the same as for the case m = 1, except that the bootstrap is applied to a series y containing blocks of length m, where each block of y is defined as $y[t] = (x[t], \ldots, x[t-m+1])$. Therefore, for the block of blocks bootstrap the first argument of statistic is given by a n x m matrix yb, where each row of yb contains one bootstrapped basic block observation $y[t]$ (n is the number of observations in x).

Note, that for statistics which are functions of the empirical m-dimensional marginal (m > 1) only this procedure yields asymptotically valid bootstrap estimates. The case m = 1 may only be used for symmetric statistics (i.e., for statistics which are invariant under permutations of x). tsboot does not implement the block of blocks bootstrap, and, therefore, the first example in tsboot yields inconsistent estimates.

For consistency, the (mean) block length b should grow with n at an appropriate rate. If b is not given, then a default growth rate of const * n^(1/3) is used. This rate is "optimal" under certain conditions (see the references for more details). However, in general the growth rate depends on the specific properties of the data generation process. A default value for const has been determined by a Monte Carlo simulation using a Gaussian AR(1) process (AR(1)-parameter of 0.5, 500 observations). const has been chosen such that the mean square error for the bootstrap estimate of the variance of the empirical mean is minimized.

Note, that the computationally intensive parts are fully implemented in C which makes tsbootstrap about 10 to 30 times faster than tsboot.

Missing values are not allowed.

There is a special print method for objects of class "resample.statistic" which by default uses max(3, getOption("digits") - 3) digits to format real numbers.

Value

If `statistic` is `NULL`, then it returns a matrix or time series with `nb` columns and `length(x)` rows containing the bootstrap data. Each column contains one bootstrap sample.

If `statistic` is given, then a list of class `"resample.statistic"` with the following elements is returned:

`statistic`	the results of applying `statistic` to each of the simulated time series.
`orig.statistic`	the results of applying `statistic` to the original series.
`bias`	the bootstrap estimate of the bias of `statistic`.
`se`	the bootstrap estimate of the standard error of `statistic`.
`call`	the original call of `tsbootstrap`.

Author(s)

A. Trapletti

References

H. R. Kuensch (1989): The Jackknife and the Bootstrap for General Stationary Observations. *The Annals of Statistics* **17**, 1217–1241.

D. N. Politis and J. P. Romano (1994): The Stationary Bootstrap. *Journal of the American Statistical Association* **89**, 1303–1313.

See Also

`sample`, `surrogate`, `tsboot`

Examples

```
n <- 500  # Generate AR(1) process
a <- 0.6
e <- rnorm(n+100)
x <- double(n+100)
x[1] <- rnorm(1)
for(i in 2:(n+100)) {
  x[i] <- a * x[i-1] + e[i]
}
x <- ts(x[-(1:100)])

tsbootstrap(x, nb=500, statistic=mean)

# Asymptotic formula for the std. error of the mean
sqrt(1/(n*(1-a)^2))

acflag1 <- function(x)
{
  xo <- c(x[,1], x[1,2])
  xm <- mean(xo)
  return(mean((x[,1]-xm)*(x[,2]-xm))/mean((xo-xm)^2))
```

```
}

tsbootstrap(x, nb=500, statistic=acflag1, m=2)

# Asymptotic formula for the std. error of the acf at lag one
sqrt(((1+a^2)-2*a^2)/n)
```

`USeconomic` *U.S. Economic Variables*

Description

This is the E.3 example data set of Luetkepohl (1991).

Usage

```
data(USeconomic)
```

Format

4 univariate time series `M1`, `GNP`, `rs`, and `rl` and the joint series `USeconomic` containing the logarithm of `M1`, the logarithm of `GNP`, `rs`, and `rl`.

Details

It contains seasonally adjusted real U.S. money `M1` and GNP in 1982 Dollars; discount rate on 91-Day treasury bills `rs` and yield on long-term treasury bonds `rl`.

Source

Luetkepohl, H. (1991): *Introduction to Multiple Time Series Analysis*. Springer Verlag, NY, 500–503.

`white.test` *White Neural Network Test for Nonlinearity*

Description

Generically computes the White neural network test for neglected nonlinearity either for the time series `x` or the regression `y~x`.

Usage

```
## S3 method for class 'ts'
white.test(x, lag = 1, qstar = 2, q = 10, range = 4,
            type = c("Chisq","F"), scale = TRUE, ...)
## Default S3 method:
white.test(x, y, qstar = 2, q = 10, range = 4,
            type = c("Chisq","F"), scale = TRUE, ...)
```

Arguments

x	a numeric vector, matrix, or time series.
y	a numeric vector.
lag	an integer which specifies the model order in terms of lags.
q	an integer representing the number of phantom hidden units used to compute the test statistic.
qstar	the test is conducted using qstar principal components of the phantom hidden units. The first principal component is omitted since in most cases it appears to be collinear with the input vector of lagged variables. This strategy preserves power while still conserving degrees of freedom.
range	the input to hidden unit weights are initialized uniformly over [-range/2, range/2].
type	a string indicating whether the Chi-Squared test or the F-test is computed. Valid types are "Chisq" and "F".
scale	a logical indicating whether the data should be scaled before computing the test statistic. The default arguments to scale are used.
...	further arguments to be passed from or to methods.

Details

The null is the hypotheses of linearity in "mean". This type of test is consistent against arbitrary nonlinearity in mean. If type equals "F", then the F-statistic instead of the Chi-Squared statistic is used in analogy to the classical linear regression.

Missing values are not allowed.

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.
method	a character string indicating what type of test was performed.
parameter	a list containing the additional parameters used to compute the test statistic.
data.name	a character string giving the name of the data.
arguments	additional arguments used to compute the test statistic.

Author(s)

A. Trapletti

References

T. H. Lee, H. White, and C. W. J. Granger (1993): Testing for neglected nonlinearity in time series models. *Journal of Econometrics* **56**, 269-290.

See Also

terasvirta.test

Examples

```
n <- 1000

x <- runif(1000, -1, 1)  # Non-linear in ``mean'' regression
y <- x^2 - x^3 + 0.1*rnorm(x)
white.test(x, y)

## Is the polynomial of order 2 misspecified?
white.test(cbind(x,x^2,x^3), y)

## Generate time series which is nonlinear in ``mean''
x[1] <- 0.0
for(i in (2:n)) {
  x[i] <- 0.4*x[i-1] + tanh(x[i-1]) + rnorm(1, sd=0.5)
}
x <- as.ts(x)
plot(x)
white.test(x)
```

Index