Insightful™
intelligence from data

S+FinMetrics
Reference Manual

May 2002

Insightful Corporation
Seattle, Washington

CONTENTS

Contents

S+FINMETRICS FUNCTION INDEX BY CATEGORY

1

FUNCTION INDEX

This following list contains the main functions in S+FinMetrics organized by high-level categories. To use this list, look for the category of function you need and locate a function that suits your purposes. You can find complete documentation for the functions in the help files of this manual or in the online S+FinMetrics help system.

Note

This function list does not include most support functions and the usual method functions for each class, such as the `residuals`, `fitted.values`, `coef`, `vcov`, `print`, `summary` and `plot` methods.

Time/Date Utility Functions

- `days.count` Calculate the number of days between two dates according to usual business rules.

- `is.weekday` Check whether a given date corresponds to a weekday.

- `is.weekend` Check whether a given date corresponds to a weekend date.

- `is.bizday` Check whether a given date corresponds to a business day.

- `imm.dates` Generate IMM dates.

Time Series Manipulation and Interpolation

- `tslag` Create lagged or leading matrices of selected orders.

- `pdl` Create polynomial distributed lags.

- `interpNA` Missing value interpolation using previous value, next value, nearest value, linear spline, or cubic spline.

- `disaggregate` Time series disaggregation/ distribution from low frequency to high frequency.

	• getReturns	Compute continuous, discrete, or overlapping returns.
	• hpfilter	Hodrick-Prescott filter.
Time Series Trellis Plotting Functions	• qqPlot	Trellis QQ-plot for Gaussian, student-t, double exponential, and generalized extreme value distributions.
	• seriesPlot	Trellis plot for univariate or multivariate time series, in one panel or multiple panels.
	• residualPlot	Trellis plot for modeling residuals with confidence band.
	• histPlot	Trellis histogram for univariate or multivariate data.
Summary Statistics and Test Statistics	• colSkewness	Compute the sample skewness column-wise for a given matrix.
	• colKurtosis	Compute the sample kurtosis column-wise for a given matrix.
	• colCumsums	Compute the cumulative sums column-wise for a given matrix.
	• summaryStats	Compute selected empirical quantile and sample moments.
	• SMA	Simple moving average.
	• EWMA	Exponential weighted moving average.
	• EWMA.cov	Exponential weighted covariance estimate.
	• rollVar	Rolling sample variance.
	• rollMax	Rolling sample maximum.
	• rollMin	Rolling sample minimum.
	• asymp.var	Compute the asymptotic or long run variance/covariance matrix.

	• normalTest	Test for normality using the Shapiro-Wilks or Jarque-Bera test.
	• autocorTest	Test for auto-correlation using the Ljung-Box, Box-Pierce or LM test.
	• archTest	LM test for ARCH effects.
Dynamic Ordinary Least Squares	• collinearTest	Test for multi-collinearity using the condition number or variance inflation factor.
	• OLS	Ordinary least squares with support for time series objects, autoregressive terms, and distributed lags.
	• vcov.OLS	Obtain the classical, HC, or HAC covariance matrix from a fitted OLS object.
	• IC.OLS	Compute the selected information criterion from a fitted OLS object.
	• heteroTest	Test for heteroskedasticity using the White or LM test.
	• waldTest	Perform the linear Wald test of fitted OLS coefficients.
Rolling Estimation	• roll	Perform generic rolling estimation.
	• rollOLS	Perform rolling OLS estimation.
	• predict.rollOLS	Perform out-of-sample prediction from a fitted rollOLS object.
	• RLS	Recursive least squares estimation.
	• cusumTest	Perform a CUSUM or CUSUMS tests based on a fitted RLS object.
Seemingly Unrelated Regression	• SUR	Linear SUR model estimation.
	• NLSUR	Nonlinear SUR model estimation.

Autoregression and Vector Autoregression

- `arima.rob`   Robust REG-ARIMA modeling and prediction.

- `VAR`   Classical vector autoregression with automatic model selection.

- `BVAR`   Bayesian vector autoregression model estimation.

- `IC.VAR`   Compute the selected information criterion given a fitted `VAR` object.

- `simulate.VAR`   Simulate from a VAR model.

- `predict.VAR`   Predict from a fitted `VAR` object.

- `cpredict.VAR`   Perform conditional prediction from a fitted `VAR` object.

- `predict.BVAR`   Predict from a fitted `BVAR` object.

- `cpredict.BVAR`   Perform conditional prediction from a fitted `BVAR` object.

- `impRes`   Compute and plot the impulse response function from a fitted `VAR` or `BVAR` object.

- `fevDec`   Compute and plot the forecast error variance decomposition from a fitted `VAR` or `BVAR` object.

- `powMat`   Return an array of powers of a square matrix.

- `arma2ma`   Moving average representation of an ARMA model.

- `VAR.ar2ma`   Return the vector moving average representation given the vector autoregression representation.

Unit Root, Cointegration and Vector Error Correction Model (VECM)	• unitroot	Perform unit root testing using the ADF or Phillips-Perron test.
	• stationaryTest	Test for stationarity against unit root or long memory.
	• quntiroot	Quantile from the asymptotic distribution of the test statistic for unit root.
	• punitroot	CDF from the asymptotic distribution of the test statistic for unit root.
	• coint	Perform the Johansen MLE test for co-integration.
	• qcoint	Quantile from the asymptotic distribution of the test statistic for cointegration.
	• pcoint	CDF from the asymptotic distribution of the test statistic for cointegration.
	• VECM	Estimate a VECM model based a fitted coint object.
	• predict.VECM	Predict from a fitted VECM object.
GARCH Volatility Modeling	• garch	Univariate GARCH modeling, which includes GARCH, EGARCH, PGARCH, TGARCH, TWO.COMP, GARCH-M and leverage effects.
	• predict.garch	Predict from a fitted univariate GARCH model.
	• mgarch	Multivariate GARCH modeling, which includes DVEC, BEKK, CCC, PRCOMP and univariate-based GARCH models.
	• predict.mgarch	Predict from a fitted multivariate GARCH model.

	• fgarch	Long memory or fractionally integrated GARCH/EGARCH modeling.
	• predict.fgarch	Predict from a fitted long memory GARCH model.
Long Memory Modeling	• rosTest	Modified rescaled range, or range over standard deviation test for long memory.
	• gphTest	GPH semi-parametric test for long memory.
	• d.ros	Estimate fractional integration parameter d using rescaled range.
	• d.pgram	Estimate fractional integration parameter d using log-periodogram regression.
	• d.whittle	Estimate fractional integration parameter d using the Whittle method.
	• FAR	Fractional autoregressive model estimation with automatic model selection.
	• FARIMA	Fractional ARIMA model estimation with automatic model selection.
	• FRIMA.spec	Compute the theoretical spectrum for a given FARIMA model.
	• FARIMA.d2ar	Compute an AR representation of a fractional noise.
	• acf.FARIMA	Compute the ACF of a given FARIMA model.
	• IC.FARIMA	Compute the selected information criterion given a FARIMA object.
	• predict.FARIMA	Predict from a fitted FARIMA object.

•	`simulate.FARIMA`	Simulate from a given FARIMA model.
•	`SEMIFAR`	Estimate a semi-parametric FARIMA model.
•	`predict.SEMIFAR`	Predict from a fitted `SEMIFAR` object.

Technical Analysis and Moving Average Operators

Price Indicators

•	`TA.Bollinger`	Bollinger band.
•	`TA.medprice`	Median price.
•	`TA.typicalPrice`	Typical price.
•	`TA.wclose`	Weighted close.

Momentum Indicators

•	`TA.accel`	Acceleration.
•	`TA.momentum`	Momentum.
•	`TA.macd`	Moving average convergence divergence.
•	`TA.roc`	Price rate of change.
•	`TA.rsi`	Relative strength index.
•	`TA.stochastic`	Stochastic oscillator.
•	`TA.williamsr`	Williams' %R.
•	`TA.williamsad`	Williams' accumulation/distribution.

Volatility Indicators

•	`TA.adoscillator`	Accumulation/distribution oscillator.
•	`TA.chaikinv`	Chaikin's volatility.
•	`TA.garmanKlass`	Garman-Klass estimator of volatility.

Volume Indicators

•	`TA.adi`	Accumulation/distribution indicator.
•	`TA.chaikino`	Chaikin's oscillator.
•	`TA.nvi`	Negative volume index.

- `TA.pvi`   Positive volume index.
- `TA.obv`   On-balance volume.
- `TA.pvtrend`   Price-volume trend.

Moving Average Operators for Tick-by-Tick Data

- `iEMA`   Inhomogeneous exponential moving average.
- `iMA`   Inhomogeneous moving average.
- `iMNorm`   Inhomogeneous moving norm.
- `iMVar`   Inhomogeneous moving variance.
- `iMSD`   Inhomogeneous moving standard deviation.
- `iMSkewness`   Inhomogeneous moving skewness.
- `iMKurtosis`   Inhomogeneous moving kurtosis.
- `iMCor`   Inhomogeneous moving correlation.
- `iDiff`   Inhomogeneous differential operator.
- `iEMA.kernel`   Kernel function of `iEMA` operators.
- `iMA.kernel`   Kernel function of `iMA` operators.

Fixed Income Analytics

- `bond.billprice`   Compute the price of Treasury bills given a quoted yield on a bank discount basis.
- `bond.discount`   Compute the discount function given different types of input.
- `bond.forward`   Compute the forward rate given different types of input.
- `bond.spot`   Compute the spot (zero-coupon) rate given different types of input.

	• term.struct	Perform term structure interpolation using the quadratic spline, cubic spline, smoothing spline, Nelson-Siegel function, or Nelson-Siegel-Svensson function.
	• predict.term.struct	Extrapolate from a fitted term.struct object.

Statistical Multi-Factor Model

	• mfactor	Perform a multi-factor model estimation using PCA or asymptotic PCA, with automatic model selection.
	• mimic	Create factor-mimicking portfolios from a fitted mfactor object.

Classical Extreme Value Theory

	• gev	Fit a generalized extreme value distribution to block maxima data.
	• gumbel	Fit a Gumbel distribution to block maxima data.
	• pgev	CDF of generalized extreme value distribution.
	• qgev	Quantiles of generalized extreme value distribution.
	• rgev	Random number generation from generalized extreme value distribution.
	• dgev	Density function of generalized extreme value distribution.
	• rlevel.gev	Calculate the k-block return level and 95% confidence interval based on a GEV model for block maxima.
	• gpd	Fit a generalized Pareto distribution to excesses over a high threshold.

- `gpd.sfall`   Calculate the expected shortfall (tail conditional expectation) estimates and confidence intervals for high quantiles above the threshold based on a GPD model.

- `gpd.q`   Calculate the quantile estimates and confidence intervals for high quantiles based on a GPD model.

- `shape`   Calculate and plot how the estimate of GPD shape parameter varies with threshold or number of extremes.

- `quant`   Calculate and plot how the estimate of a high quantile in a GPD model varies with threshold or number of extremes.

- `riskmeasures`   Make a rapid calculation of point estimates of prescribed quantiles and expected shortfalls using the output of the function `gpd`.

- `pgpd`   CDF of the generalized Pareto distribution.

- `qgpd`   Quantiles of the generalized Pareto distribution.

- `rgpd`   Random number generation from the generalized Pareto distribution.

- `dgpd`   Density function of the generalized Pareto distribution.

- `gpdbiv`   Bivariate POT (peaks over thresholds) analysis.

- `pot`   Augmented POT with point processes.

- `hill`   Hill estimate of the tail index of heavy-tailed data.

- `exindex`   Estimate the extremal index using the blocks method.

- `records`   Calculate the record development.

•	`emplot`	Generate a plot of the empirical distribution function of a sample.
•	`qplot`	QQ-plot for threshold data against the exponential distribution or the generalized Pareto distribution.
•	`meplot`	Calculate and plot sample mean excesses over increasing thresholds.

Extreme Value Analysis with Statistical Copula Estimation

•	`gpd.tail`	Fit a generalized Pareto distribution to excesses on two tails (standard POT analysis).
•	`shape.plot`	Calculate and plot how the shape parameter of a GPD varies with thresholds.
•	`gpd.lmom`	Compute L-moment parameter estimates for GPD.
•	`gpd.ml`	Compute MLE parameter estimate for GPD.
•	`gpd.1p`	Semi-parametric estimation of CDF based on a GPD model (one tail).
•	`gpd.2p`	Semi-parametric estimation of CDF based on a GPD model (two tails).
•	`gpd.1q`	Semi-parametric estimation of the quantile based on a GPD model (one tail).
•	`gpd.2q`	Semi-parametric estimation of the quantile based on a GPD model (two tails).
•	`gev.lmom`	Compute L-moment parameter estimates for GEV.
•	`gev.mix1`	Compute MIX1 parameter estimates for GEV.
•	`gev.mix2`	Compute MIX2 parameter estimates for GEV.

- `sample.LMOM`   Compute unbiased estimates of mean, second *L*-moment, *L*-skewness, and *L*-kurtosis.

- `PlotPos.LMOM`   Compute plotting position estimates of sample *L*-moments

- `bivd`   Create an object representing a bivariate distribution with arbitrary marginals.

- `pbivd`   CDF of an arbitrary bivariate distribution.

- `dbivd`   Density of an arbitrary bivariate distribution.

- `rbivd`   Random variate generation of an arbitrary bivariate distribution.

- `gpdjoint.1p`   Empirical and semi-parametric estimate of bivariate joint CDF (one tail).

- `gpdjoint.2p`   Empirical and semi-parametric estimate of bivariate joint CDF (two tails).

- `VaR.exp.portf`   Calculate the Value-at-Risk of a two-asset portfolio based on the copula parameters and fitted GPD models.

- `VaR.exp.sim`   Calculate the Value-at-Risk and expected shortfall of a two-asset portfolio by simulation methods.

- `copula`   Create a copula object.

- `archm.copula`   Create an Archimedean copula object.

- `ev.copula`   Create an Extreme Value copula object.

- `empirical.copula` Create an empirical copula object.

- `dcopula`   Density of two random variables with uniform marginals and a joint CDF given by a `copula` object.

- pcopula   CDF of two random variables with uniform marginals and a joint CDF given by a `copula` object.

- rcopula   Random number generation of two random variables with uniform marginals and a joint CDF given by a `copula` object.

- tail.index   Compute the tail dependence index for a parametric or empirical copula.

- Afunc   Calculate the dependence function for an extreme value copula.

- PHI   calculate the Archimedean generator function for an Archimedean copula.

- Spearmans.rho   compute Spearman's rho for a copula.

- Kendalls.tau   compute Kendall's tau for a copula.

- fit.copula   MLE parameter estimates for copulas.

- Parametric copula families include:

  bb1.copula, bb2.copula, bb3.copula, bb4.copula, bb5.copula, bb6.copula, bb7.copula, frank.copula, galambos.copula, gumbel.copula, husler.reiss.copula, joe.copula, kimeldorf.sampson.copula, normal.copula, normal.mix.copula, tawn.copula.

State Space Modeling

- CheckSsf   Return a proper state space form giving a minimal necessary representation.

- GetSsfArma   Return a state space form for an ARMA model.

- GetSsfReg   Return a state space form for a linear regression model.

- GetSsfRegArma   Return a state space form for a REGARIMA model.

- GetSsfSpline   Return a state space form for a spline model.

- `GetSsfStsm`   Return a state space form for a basic structural model.

- `KalmanFil`   Perform Kalman filtering.

- `KalmanSmo`   Perform Kalman smoothing.

- `SimSmoDraw`   Sample from the conditional density using the disturbance simulation smoother or state simulation smoother.

- `SsfCondDens`   Compute the conditional mean of smoothed disturbances or states.

- `SsfFit`   Estimate the unknown parameters of a state space model using prediction error decomposition.

- `SsfLoglike`   Compute the log-likelihood value of a state space model using prediction error decomposition.

- `SsfMomentEst`   Perform state prediction, state filtering, state smoothing, or disturbance smoothing based on a state space model.

- `SsfSim`   Simulate from a state space model.

S+FINMETRICS FUNCTION HELP FILES

2

acf.FARIMA   Theoretical Autocovariance Function of ARIMA/FARIMA
Models

DESCRIPTION

Returns the theoretical autocovariance function given a (fractional)
ARIMA model.

```
acf.FARIMA(model, lag.max, kapprox=100000)
```

REQUIRED ARGUMENTS

model : a list with the following named components: "d" which is the differ-
ence parameter and could be fractional or the FARIMA models, "ar"
is the AR coefficients, "ma" is the MA coefficients, and "sigma2" is
the residual variance. The model component of a "FARIMA" object
provides such a list.

OPTIONAL ARGUMENTS

lag.max : an integer giving the maximum lags to use for computing the auto-
covariance function.

kapprox : a large integer that determines how many Fourier frequencies to use
in evaluating the theoretical spectrum of a FARIMA model. This is
only used for FARIMA models when theoretical autocovariance func-
tions cannot be computed exactly. It is ignored for a regular ARMA
model. The default is set to 100000.

VALUE

a list containing the following components:

lags : an integer vector from 0 to lag.max.

acf : a numeric vector giving the theoretical autocovariance corresponding
to lags.

DETAILS

For FARIMA models, this function calls FARIMA.spec to evaluate the
theoretical spectrum of a FARIMA model at frequencies (1:(kapprox-1))/kapprox
(cycles/sample).

REFERENCES

Beran, J. (1994). *Statistics for Long Memory Processes*, Chapman
and Hall, New York.

Bhansali, R. J., and Kokoszka, P. S. (2001). Computation of the
forecast coefficients for multistep prediction of long-range dependent
time series. *International Journal of Forecasting*, forthcoming.

McLeod, I. (1975). Derivation of the theoretical autocovariance function of autoregressive-moving average time series. *Applied Statistics*, 24(2):255-256.

SEE ALSO

FARIMA.spec, arma2ma .

EXAMPLE

```
# evaluate the theoretical ACF for FARIMA(0, 0.3, 0)
acf.FARIMA(list(d=0.3, sigma2=1), 10)
```

acfPlot   Trellis ACF/PACF Plot

DESCRIPTION

Generates a trellis object representing the autocovariance, autocorrelation or partial autocorrelation plot.

```
acfPlot(x, y, sigma=NULL, width=1.96, strip.text="",
        hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

x : a numeric vector or a `"timeSeries"` object with a numeric vector in the data slot. This is usually the lag index which appears on the horizontal axis.

y : a numeric vector or a `"timeSeries"` object with a numeric vector in the data slot. This usually contains the values for autocovariance, autocorrelation, or partial autocorrelation.

OPTIONAL ARGUMENTS

sigma : a number giving the standard error of the ACF/PACF function. If `sigma` is not `NULL`, a confidence band will be drawn for the ACF/PACF function. The default is `NULL`.

width : a number giving the width of the confidence band in units of `sigma`. This is ignored if `sigma` is `NULL`. The default is `1.96`.

strip.text : a character string which will appear in the strip of the trellis plot. The default is `""`, which draws nothing in the strip.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the trellis plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the trellis plot. The default is `FALSE`.

... : any other optional arguments that can be passed down to low level plot functions.

VALUE

an object of class `trellis`, which is automatically plotted by `print.trellis`. If this function is invoked directly without saving the returned object, a trellis device will be open to draw the plot if one is not available yet.

SEE ALSO

`acf`,`trellis.args`.

EXAMPLE

```
# plots the theoretical ACF of a fractional noise
fd.acf = acf.FARIMA(list(d=0.3, sigma2=1), lag.max=100)
acfPlot(fd.acf$lags, fd.acf$acf, strip="FARIMA(0, 0.3, 0)")
```

acf.FARIMA Theoretical Autocovariance Function of ARIMA/FARIMA Models

DESCRIPTION

Returns the theoretical autocovariance function given a (fractional) ARIMA model.

```
acf.FARIMA(model, lag.max, kapprox=100000)
```

REQUIRED ARGUMENTS

model : a list with the following named components: `"d"` which is the difference parameter and could be fractional or the FARIMA models, `"ar"` is the AR coefficients, `"ma"` is the MA coefficients, and `"sigma2"` is the residual variance. The `model` component of a `"FARIMA"` object provides such a list.

OPTIONAL ARGUMENTS

lag.max : an integer giving the maximum lags to use for computing the auto-covariance function.

kapprox : a large integer that determines how many Fourier frequencies to use in evaluating the theoretical spectrum of a FARIMA model. This is only used for FARIMA models when theoretical autocovariance functions cannot be computed exactly. It is ignored for a regular ARMA model. The default is set to `100000`.

VALUE

a list containing the following components:

lags : an integer vector from 0 to `lag.max`.

acf : a numeric vector giving the theoretical autocovariance corresponding to `lags`.

DETAILS

For FARIMA models, this function calls `FARIMA.spec` to evaluate the theoretical spectrum of a FARIMA model at frequencies `(1:(kapprox-1))/kapprox` (cycles/sample).

REFERENCES

Beran, J. (1994). *Statistics for Long Memory Processes*, Chapman and Hall, New York.

Bhansali, R. J., and Kokoszka, P. S. (2001). Computation of the forecast coefficients for multistep prediction of long-range dependent time series. *International Journal of Forecasting*, forthcoming.

McLeod, I. (1975). Derivation of the theoretical autocovariance function of autoregressive-moving average time series. *Applied Statistics*, 24(2):255-256.

SEE ALSO

FARIMA.spec, arma2ma .

EXAMPLE

```
# evaluate the theoretical ACF for FARIMA(0, 0.3, 0)
acf.FARIMA(list(d=0.3, sigma2=1), 10)
```

acfPlot Trellis ACF/PACF Plot

DESCRIPTION

Generates a trellis object representing the autocovariance, autocorre-
lation or partial autocorrelation plot.

```
acfPlot(x, y, sigma=NULL, width=1.96, strip.text="",
        hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

x : a numeric vector or a `"timeSeries"` object with a numeric vector
in the data slot. This is usually the lag index which appears on the
horizontal axis.

y : a numeric vector or a `"timeSeries"` object with a numeric vector in
the data slot. This usually contains the values for autocovariance,
autocorrelation, or partial autocorrelation.

OPTIONAL ARGUMENTS

sigma : a number giving the standard error of the ACF/PACF function. If
`sigma` is not `NULL`, a confidence band will be drawn for the ACF/PACF
function. The default is `NULL`.

width : a number giving the width of the confidence band in units of `sigma`.
This is ignored if `sigma` is `NULL`. The default is `1.96`.

strip.text : a character string which will appear in the strip of the trellis
plot. The default is `""`, which draws nothing in the strip.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the trellis
plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the trellis plot.
The default is `FALSE`.

... : any other optional arguments that can be passed down to low level
plot functions.

VALUE

an object of class `trellis`, which is automatically plotted by `print.trellis`.
If this function is invoked directly without saving the returned object,
a trellis device will be open to draw the plot if one is not available yet.

SEE ALSO

`acf`,`trellis.args` .

EXAMPLE

```
# plots the theoretical ACF of a fractional noise
fd.acf = acf.FARIMA(list(d=0.3, sigma2=1), lag.max=100)
acfPlot(fd.acf$lags, fd.acf$acf, strip="FARIMA(0, 0.3, 0)")
```

ALLTOPAR GARCH/MGARCH Parameter Mapping

DESCRIPTION

Maps all the GARCH/MGARCH parameters to the parameters to be estimated. It is not supposed to be called by the users directly.

```
ALLTOPAR(ALLPAR, PINDEX, NINDEX)
```

archbhhh Estimate GARCH/MGARCH Models Using BHHH Algorithm

DESCRIPTION

This function is called by `garch()` and `mgarch()` to conduct maximum likelihood estimation using BHHH algorithm. It is not supposed to be called by the users directly.

```
archbhhh(ALLPAR, PINDEX, NINDEX, model=NULL, DEV=6, IDIS=0, ID24=F,
    KONST=T, LEV=0, MF=0, IFG=0, ny=NULL, NH=1, atype=0, btype=0,
    ctype=0, symm=F, y=NULL, x=NULL, z=NULL, NXEXO=0,
    NZEXO=0, Yvar=NULL, Ycor=NULL, control=NULL, CDIST=NULL)
```

archm.copula.object   Archimedean Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"archm.copula"`, which inherits from `copula` class.

GENERATION

Sub-class objects of class Archimedean copula class can be constructed from the `archm.copula(family, param)` function.

STRUCTURE

The following slots must be present in a legitimate `"archm.copula"` class object:

`parameters` : a vector with values of the parameters for this particular instance of the object copula (the length of this vector is equal to the number of parameters allowed for this particular family).

`param.names` : typical greek letters that are used for the parameters (such as `delta` or `theta`).

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s).

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message`: a message specifying the name of the parametric copula family, and the copula sub-class, such as Archimedean copula, if applicable.

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object`,`archm.copula` .

archm.copula Construction of a Specific Archimedean Copula Class Object

DESCRIPTION

Constructs a certain Archimedean Copula class object with its corresponding parameter vector.

```
archm.copula(family="BB1", param=c(0.8, 1.43))
```

REQUIRED ARGUMENTS

family: a character string defining which Archimedean copula class to generate.

param: a numeric vector specifying the value for the parameters of this particular Archimedean copula class.

VALUE

an object of certain Archimedean Copula class specified by `family` argument (which inherits from `copula` and `archm.copula` classes).

DETAILS

Each Archimedean copula sub-class can also be constructed directly.

SEE ALSO

`archm.copula.object`,`ev.copula` .

archTest   LM Test for ARCH Effects

DESCRIPTION

Returns the test statistics for ARCH effects using the LM test.

This function is an S Version 3 generic (see `Methods`); method functions can be written to handle specific classes of data. If the input is a time series, then the ARCH test is applied on this time series; if the input is a fitted model object, then the ARCH test is applied on the `"residuals"` component of the object if it is present.

```
archTest(x, lag.n=NULL, sigma=NULL, na.rm=F)
```

REQUIRED ARGUMENTS

x : a numeric vector, or a `"timeSeries"` object with a numeric vector in the data slot, or a list with a `"residuals"` component such as most fitted model objects.

OPTIONAL ARGUMENTS

lag.n : an integer specifying the maximum number of lags to use in the regression. By default, it is set to `10 * log10(n)` where `n` is the sample size.

sigma : a numeric vector, or a `"timeSeries"` object with a numeric vector in the data slot, which can be the conditional standard deviation of the input data. If `sigma` is not `NULL`, it will be used to standardize the input data.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computing the test statistic. The default is `FALSE`.

VALUE

an object of class `"archTest"`, which contains the following components:

stat : the test statistic, usually computed as `n*R2`, where `n` is the sample size, and `R2` is the r-squared of the auxiliary regression.

distribution : a character string giving the distribution of the test statistic under the null hypothesis.

parameters : the degree of freedom of `distribution` under the null hypothesis.

p.value : the probability that the null hypothesis is true.

n : an integer giving the sample size.

na : an integer giving the number of missing values in the input if any.

REFERENCES

Engle, R. F. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4):987-1007.

SEE ALSO

`autocorTest,normalTest` .

EXAMPLE

`archTest(dell.s, lag=12)`

arima.rob.object Robust REGARIMA Model and Outliers Detection
Objects

DESCRIPTION

> These are objects of class `"arima.rob"` which represent the robust fit
> of a regression model with ARIMA errors. It also contains information
> about the detected outliers.

GENERATION

> This class of objects is returned from the `arima.rob` function.

METHODS

> `coef`, `formula`, `outliers`, `plot`, `predict`, `print`, `summary`.

STRUCTURE

> The following components must be included in a legitimate `"arima.rob"`
> object:

VALUE

> x: the model matrix.

> y: the response variable.

> model: a list with the following named components: `"freq"` which is the fre-
> quency of the original data, `"sfreq"` which is the seasonal frequency
> of the original data, `"d"` which is the number of regular differences,
> `"sd"` which is the number of seasonal differences, `"ar"` which is the es-
> timated AR coefficients, `"ma"` which is the estimated MA coefficients,
> `"sma"` which is the seasonal MA coefficient if estimated.

regcoef: the estimates of regression coefficients.

regcoef.cov: the estimated covariance matrix of the regression coefficients.

> innov: the estimated innovations.

innov.acf: a series whose autocorrelations or partial autocorrelations are the
robust estimates of the innovation autocorrelations or partial autocor-
relations.

regresid: the estimated regression residuals cleaned of additive outliers by the
robust filter.

regresid.acf: a series whose autocorrelations or partial autocorrelations are
the robust estimates of the autocorrelations or partial autocorrelations
of the differenced regression residuals.

`sigma.innov:` a robust estimate of the innovation scale.

`sigma.regresid:` an estimate of the scale of the differenced regression residuals.

`sigma.first:` the first estimate of the innovation scale based only on the scale of the differenced model and the ARMA parameters.

`tuning.c:` the bandwidth of the robust filter.

`y.robust:` the response series cleaned of outliers by the robust filter.

`y.cleaned:` the response series cleaned of additive outliers and level shifts after the outliers detection procedure.

`predict.error:` the fitted and predicted regression errors.

`predict.scales:` the standard deviations of the fitted and predicted regression errors.

`n.predict:` the number of predicted observations, which is equal to the `n.predict` argument passed to the `arima.rob` function that produced the `"arima.rob"` object.

`tauef:` the inverse of the estimated efficiency factor of the tau-estimate with respect to the LS-estimate.

`inf:` information about the outcome of the last optimization procedure: `inf=1` indicates that the procedure converged, and `inf=0` that the procedure did not converge.

`innov.outlier:` logical flag, the same as the `innov.outlier` argument passed to the `arima.rob` function that produced the `"arima.rob"` object.

`outliers:` an object of class `"outliers"`, which contains all the detected outliers (and level shifts).

`outliers.iter:` optionally a list of objects of class `"outliers"`, if the `iter` argument passed to the `arima.rob` function that produced the `"arima.rob"` object is non-zero.

`n0:` the number of missing innovations at the beginning.

`call:` an image of the call that produced the object, but with the arguments all named and with the actual formula included as the `formula` argument.

`assign:` the same as the `assign` component of an `"lm"` object.

`contrasts:` the same as the `contrasts` component of an `"lm"` object.

`terms:` the same as the `terms` component of an `"lm"` object.

rank: the same as the `rank` component of an `"lm"` object.

SEE ALSO

`arima.rob`,`outliers` ,`outliers.object` .

arima.rob Robust Fit of a REGARIMA Model and Outliers Detection

DESCRIPTION

Returns an object of class `"arima.rob"` that represents a robust fit of a linear regression model with ARIMA errors using a filtered tau-estimate. The error model may have seasonal differences and one seasonal moving average parameter. It also returns the detected outliers and level shifts.

```
arima.rob(formula, data, contrasts=NULL, start=NULL, end=NULL,
     p=0, q=0, d=0, sd=0, freq=1, sfreq=NULL, sma=F,
     max.p=NULL, auto.ar=F, n.predict=20, tol=10^ (-6),
     max.fcal=2000, innov.outlier=F, critv=NULL, iter=F)
```

REQUIRED ARGUMENTS

formula: a `formula` object, with the response on the left of a $\sim$ operator, and the terms, separated by + operators, on the right.

OPTIONAL ARGUMENTS

data: a data frame or a `"timeSeries"` object with a data frame in the data slot, which is used to interpret the variables named in `formula`. If this is missing, then the variables in `formula` should be on the search list. Missing values are not allowed.

contrasts: the same as the `contrasts` argument for `lm` function.

start : a character string which can be passed to `timeDate` function to specify the starting date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

end : a character string which can be passed to `timeDate` function to specify the ending date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

p: the autoregressive order of the errors model. The default is 0.

q: the moving average order of the errors model. The default is 0.

d: the number of regular differences in the ARIMA model. It must be 0, 1 or 2. The default is 0.

sd: the number of seasonal differences. It must be 0, 1 or 2. The default is 0.

freq: the frequency of `data`. The default is 1.

sfreq: the seasonality frequency of `data`. If `NULL`, it is set to be equal to `freq`. The default is `NULL`.

sma: logical flag: if `TRUE`, the errors model includes a seasonal moving average parameter. The default is `FALSE`.

auto.ar: logical flag: If `TRUE` an AR(p) model is selected automatically using a robust AIC criterion. The default is `FALSE`.

max.p: the maximum order of the autoregressive stationary model that approximates the ARMA stationary model. If `NULL`, `max.p=max(p+q,5)`. If q=0, then `max.p` is not necessary. The default is `NULL`.

n.predict: the maximum number of future periods for which we wish to compute the predictions. The default is 20.

innov.outlier: logical flag: if `TRUE`, the function `arima.rob` looks for innovation outliers in addition to additive outliers and level shifts; otherwise, `arima.rob` only looks for additive outliers and level shifts. The default is `FALSE`.

critv: the critical value for detecting outliers. If `NULL`, it assumes the following default values: `critv=3` if the length of the time series is less than 200; `critv=3.5` if it is between 200 and 500, and `critv=4` if it is greater than 500.

iter: a logical flag or the number of iterations to execute `arima.rob` with.

VALUE

an object of class `"arima.rob"` representing the fit and the outliers detected. See `arima.rob.object` for details of the components of the object.

WARNING

When either `d` or `sd` is greater than zero, the interpretation of the intercept in the `formula` is different from its usual interpretation: it represents the coefficient of the lowest order power of the time trend which can be identified. For example, if `d=2` and `sd=0`, the intercept represents the coefficient of the term `t^2`.

REFERENCES

Bianco, A., Garcia Ben, M., Martinez, E., and Yohai, V. (1996). Robust procedures for regression models with ARIMA errors. *COMPSTAT 96, Proceedings in Computational Statistics*. Ed. Albert Prat, pages. 27-38. Physica-Verlag, Heidelberg.

Bianco, A., Garcia Ben, M., Martinez, E., and Yohai, V. (1997). Outlier detection in regression models with ARIMA errors using robust estimates. mimeo.

Chang, I., Tiao, G. C., and Chen, C. (1988). Estimation of time series parameters in the presence of outliers. *Technometrics*, 30:193-204.

Martin, R. D., Samarov, A., and Vandaele, W. (1983). Robust methods for ARIMA models. in *Applied Time Series Analysis of Economic Data*, E. Zellner, ed.

Yohai, V. Y., and Zamar, R. H. (1988). High breakdown-point estimates of regression by means of the minimization of an efficient scale. *Journal of the American Statistical Association*, 83:406-413.

SEE ALSO

> `arima.rob.object`.

EXAMPLE

> `frip.rr = arima.rob(log(frip)~1, p=2, d=1)`

arma2ma Moving Average Representation of an ARMA Model

DESCRIPTION

Returns the moving average representation of an autoregressive moving average (ARMA) model.

```
arma2ma(model, lag.max)
```

REQUIRED ARGUMENTS

model : a list with the following components: `ar` which gives the autoregressive coefficients, and `ma` which gives the moving average coefficients.

lag.max : an integer which specifies the order of the moving average representation.

VALUE

a numeric vector with length `lag.max + 1` giving the moving average representation, with the first element being one.

DETAILS

The specification and representation of the moving average part is different from that required by `arima.mle`. In particular, the signs are the opposite.

SEE ALSO

`FARIMA.d2ar`,`VAR.ar2ma` .

EXAMPLE

```
# returns MA(10) representation of an ARMA(1,1) model
arma2ma(list(ar=0.8, ma=0.1), 10)
```

array2list Conversion Between an Array and a List

DESCRIPTION

Converts a 3D array to a list of length L with logical matrix, vector, or scalar components.

```
array2list(x, N, L=NULL, type=1, symm=F, N1=N)
list2array(x, N, M, type=1, N1=N)
```

REQUIRED ARGUMENTS

x: a 3D array.

N: dimension of the matrices or vectors.

L: the length of the list. The default is the number of rows M of x.

type: a number between 0 and 4. 0 indicates the result is a list of N by N matrices; 1 indicates a list of lower-triangular (if symm=TRUE then symmetric) matrices; 2 indicates a list of diagonal matrices; 3 indicates a list of vectors; 4 indicates a list of scalars.

symm: a logical flag. It is only used when type=1.

N1: the dimension for exogenous variables.

VALUE

a list specified as above.

SEE ALSO

list2mat, mat2vec .

asymp.var Long Run or Asymptotic Variance-Covariance Matrix

DESCRIPTION

Returns the long run or asymptotic variance-covariance matrix for univariate or multivariate time series data.

```
asymp.var(x, bandwidth, window="bartlett", na.rm=F)
```

REQUIRED ARGUMENTS

x : a vector, matrix, or any other rectangular data object, with each column treated as one variable.

OPTIONAL ARGUMENTS

bandwidth : a positive integer that specifies the bandwidth to be used. It may not be greater than the number of rows in x.

window : a character string that specifies the type of window to be used. Currently the only valid choices are "Bartlett" for Bartlett's triangular window, or "rectangular" for a rectangular window. The default is "Bartlett".

na.rm : a logical flag: if TRUE, missing values in x will be removed before computing the long run variance; if FALSE, an error message will be printed if there are missing values in x. The default is FALSE.

VALUE

a numeric value which is the long run variance if x is a vector or has only one column, or a matrix which is the long run covariance matrix if x is rectangular.

REFERENCES

Newey, W. K., and West, K. D. (1987). A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55(3):703-08.

EXAMPLE

```
band = 4*floor((length(dell.s)/100)^0.25)
asymp.var(dell.s, band)
```

autocorTest   Test for Autocorrelation in Time Series Data

DESCRIPTION

Returns the statistics for various tests for serial correlation given a univariate or multivariate time series data.

This function is an S Version 3 generic (see `Methods`); method functions can be written to handle specific classes of data. If the input is a time series, then the test is applied on this time series; if the input is a fitted model object, then the test is applied on the `"residuals"` component of the object if it is present.

```
autocorTest(x, lag.n=1, method="lb", bycol=T, na.rm=F)
```

REQUIRED ARGUMENTS

x : a vector, matrix, data frame, or a `"timeSeries"` object with a numeric object in the data slot, or a list with a `"residuals"` component such as most fitted model objects.

OPTIONAL ARGUMENTS

lag.n : an integer specifying the maximum number of lags to use in computing the autocorrelation. By default, it is set to `10 * log10(n)` where `n` is the sample size.

method : a character string specifying the type of the test. Currently the valid choices are: `"lb"` for Ljung-Box type test, and `"bp"` for Box-Pierce type test. The default is `"lb"`.

bycol : logical flag: if `TRUE` and `x` has more than one column, then the test is applied to each column separately; if `FALSE` and `x` has more than one column, then the multivariate test is applied. The default is `TRUE`.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computing the test statistic. The default is `FALSE`.

VALUE

an object of class `"autocorTest"`, which contains the following components:

method : a character string which is the same as in the input.

statistic : the test statistics according to the chosen `method`.

distribution : a character string giving the distribution of the test statistic under the null hypothesis.

parameters : the degree of freedom of `distribution` under the null hypothesis.

p.value : the probability that the null hypothesis is true.

n : an integer giving the sample size.

na : an integer giving the number of missing values in the input if any.

REFERENCES

Box, G. E. P., and Pierce, D. A. (1970). The distribution of residual autocorrelations in autoregressive integrated moving average models. *Journal of the American Statistical Association*, 65:1509-1526.

Hosking, J. R. M. (1980). The multivariate portmanteau statistic. *Journal of the American Statistical Association*, 75:602-608.

Ljung, G. M., and Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, 65:297-303.

SEE ALSO

`archTest,normalTest` .

EXAMPLE

`autocorTest(dell.s, lag=12)`

bb1.copula.object BB1 Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"bb1.copula"` , which inherits from `copula` and `archm.copula` classes.

GENERATION

This class of objects is constructed from the `bb1.copula(theta, delta)` function or `archm.copula` function.

METHODS

The `"bb1.copula"` class of objects currently has methods for the following generic functions:

PHI, InvPhi, PhiDer, InvPhifirstDer, InvPhisecondDer, pcopula, dcopula, rcopula, contour.plot, LAMBDA<code>,<code>tail.index, Kendalls.tau, Spearmans.rho, dcdx.

Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula`, `contour.dcopula`, `contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"bb1.copula"` class object:

`parameters` : a vector with values for the parameter `theta` (greater than 0) and `delta` (greater or equal to 1).

`param.names` : `theta` and `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (BB1 copula family), and the copula class from which it inherits (Archimedean Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object`,`archm.copula.object` ,`archm.copula` .

bb2.copula.object BB2 Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"bb2.copula"` , which inherits from `copula` and `archm.copula` classes.

GENERATION

This class of objects is constructed from the `bb2.copula(theta, delta)` function or `archm.copula` function.

METHODS

The `"bb2.copula"` class of objects currently has methods for the following generic functions:

`PHI, InvPhi, PhiDer, InvPhisecondDer, pcopula, dcopula, rcopula, contour.plot, LAMBDA<code>, <code>tail.index, Kendalls.tau, Spearmans.rho, dcdx.`

Furthermore, following functions are implemented for this class: `persp.dcopula, persp.pcopula, contour.dcopula, contour.pcopula.`

STRUCTURE

The following slots must be present in a legitimate `"bb2.copula"` class object:

`parameters` : a vector with values for the parameter `theta` (greater than 0) and `delta` (greater or equal to 1).

`param.names` : `theta` and `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (BB2 copula family), and the copula class from which it inherits (Archimedean Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object`,`archm.copula.object` ,`archm.copula` .

bb3.copula.object BB3 Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"bb3.copula"` , which inherits from `copula` and `archm.copula` classes.

GENERATION

This class of objects is constructed from the `bb3.copula(theta, delta)` function or `archm.copula` function.

METHODS

The `"bb3.copula"` class of objects currently has methods for the following generic functions:

PHI, InvPhi, PhiDer, InvPhisecondDer, pcopula, dcopula, rcopula, contour.plot, LAMBDA<code>, <code>tail.index, Kendalls.tau, Spearmans.rho, dcdx.

Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula, contour.dcopula, contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"bb3.copula"` class object:

`parameters` : a vector with values for the parameter `theta` (greater or equal to 1) and `delta` (greater than 0).

`param.names` : `theta` and `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message`: a message specifying the name of the parametric copula family (BB3 copula family), and the copula class from which it inherits (Archimedean Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object`,`archm.copula.object` ,`archm.copula` .

bb4.copula.object BB4 Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"bb4.copula"` , which inherits from `copula` class.

BB4 copula is Archimax copula type.

GENERATION

This class of objects is constructed from the `bb4.copula(theta, delta)` function.

METHODS

The `"bb4.copula"` class of objects currently has methods for the following generic functions:

`Afunc, AfirstDer, AsecondDerPhiDer, Hderiv, PHI, InvPhi, PhiDer, InvPhifirstDer, InvPhisecondDer, LAMBDA, pcopula, dcopula, rcopula, contour.plot, tail.index, Kendalls.tau, Spearmans.rho, dcdx.`

Furthermore, following functions are implemented for this class: `persp.dcopula, persp.pcopula, contour.dcopula, contour.pcopula.`

STRUCTURE

The following slots must be present in a legitimate `"bb4.copula"` class object:

`parameters` : a vector of values for the parameter(s) `theta` (greater or equal to 0) and `delta` (greater than 0).

`param.names` : `theta` and `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (BB4 copula family), and the copula class it belongs to (Archimax Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object.`

bb5.copula.object BB5 Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"bb5.copula"`, which inherits from `copula` and `ev.copula` classes.

GENERATION

This class of objects is constructed from the `bb5.copula(theta, delta)` function or `ev.copula` function.

METHODS

The `"bb5.copula"` class of objects currently has methods for the following generic functions:

`Afunc`, `AfirstDer`, `AsecondDerPhiDer`, `Hderiv`, `pcopula`, `dcopula`, `rcopula`, `contour.plot`, `tail.index`, `Kendalls.tau`, `Spearmans.rho`, `dcdx`.

Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula`, `contour.dcopula`, `contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"bb5.copula"` class object:

`parameters` : a vector of values for the parameter(s) `theta` (greater or equal to 0) and `delta` (greater than 0).

`param.names` : `theta` and `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (BB5 copula family), and the copula class from which it inherits (Extreme Value Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object`, `ev.copula.object`, `ev.copula`.

bb6.copula.object   BB6 Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"bb6.copula"` , which inherits from `copula` and `archm.copula` classes.

GENERATION

This class of objects is constructed from the `bb6.copula(theta, delta)` function or `archm.copula` function.

METHODS

The `"bb6.copula"` class of objects currently has methods for the following generic functions:

PHI, InvPhi, PhiDer, InvPhisecondDer, pcopula, dcopula, rcopula, contour.plot, LAMBDA<code>, <code>tail.index, Kendalls.tau, Spearmans.rho, dcdx.

Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula`, `contour.dcopula`, `contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"bb6.copula"` class object:

`parameters` : a vector with values for the parameter `theta` (greater or equal to 0) and `delta` (greater than 0).

`param.names` : `theta` and `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (BB6 copula family), and the copula class from which it inherits (Archimedean Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object`,`archm.copula.object` ,`archm.copula` .

bb7.copula.object BB7 Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"bb7.copula"` , which inherits from `copula` and `archm.copula` classes.

GENERATION

This class of objects is constructed from the `bb7.copula(theta, delta)` function or `archm.copula` function.

METHODS

The `"bb7.copula"` class of objects currently has methods for the following generic functions:

PHI, InvPhi, PhiDer, InvPhifirstDer, InvPhisecondDer, pcopula, dcopula, rcopula, contour.plot, LAMBDA<code>,<code>tail.index, Kendalls.tau, Spearmans.rho, dcdx.

Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula`, `contour.dcopula`, `contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"bb7.copula"` class object:

`parameters` : a vector with values for the parameter `theta` (greater or equal to 0) and `delta` (greater than 0).

`param.names` : `theta` and `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (BB7 copula family), and the copula class from which it inherits (Archimedean Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object`,`archm.copula.object` ,`archm.copula` .

berndt.dat Monthly Stock Return Data

SUMMARY
> This is a monthly `"timeSeries"` object from January 1978 to December 1987, with seventeen columns representing monthly returns of certain assets, as in Chapter 2 of Berndt (1991).

CITCRP: monthly returns of Citicorp.

CONED: monthly returns of Consolidated Edison.

CONTIL: monthly returns of Continental Illinois.

DATGEN: monthly returns of Data General.

DEC: monthly returns of Digital Equipment Company.

DELTA: monthly returns of Delta Airlines.

GENMIL: monthly returns of General Mills.

GERBER: monthly returns of Gerber.

IBM: monthly returns of International Business Machines.

MARKET: a value-weighted composite monthly returns based on transactions from the New York Stock Exchange and the American Exchange.

MOBIL: monthly returns of Mobile.

PANAM: monthly returns of Pan American Airways.

PSNH: monthly returns of Public Service of New Hampshire.

TANDY: monthly returns of Tandy.

TEXACO: monthly returns of Texaco.

WEYER: monthly returns of Weyerhauser.

RKFREE: monthly returns on 30-day U.S. Treasury bills.

SOURCE
> Berndt, E. R. (1991). *The Practice of Econometrics: Classic and Contemporary*. Addison-Wesley Publishing Co.

bhhh.control Control BHHH Parameters for GARCH/MGARCH Estimation

DESCRIPTION

Allows the users to set values affecting BHHH algorithm used for GARCH/MGARCH estimation in `garch` and `mgarch`.

```
bhhh.control(tol=0.0001, delta=0.0001, stepsize=NULL,
             n.iter=100, zigzag=F, positive=F)
```

OPTIONAL ARGUMENTS

`tol`: convergence tolerance in parameter estimation. The default is 0.0001.

`delta`: the step length for numerical derivatives. The default is 0.0001.

`stepsize`: a number specifying the stepsize in the optimization iteration. If `stepsize=0`, the routine will automatically choose the initial stepsize in the iterations. If `stepsize=d`, with `d` being an arbitrary positive number, then the routine uses `d` as the basic stepsize, and the actual stepsize in the iterations will be multiples of `d`.

`n.iter`: the maximum number of iterations in optimization. The default is 100.

`zigzag`: a logical flag: if `TRUE`, a special feature is used to handle optimization of a dimensional likelihood function.

`positive`: a logical flag: if `TRUE`, the coefficients in the variance equation of GARCH models will be constrained to be positive during optimization.

VALUE

a list containing the values used for each of the control parameters.

EXAMPLE

```
hp.garch = garch(hp.s~1, ~garch,
    control=bhhh.control(tol=0.00001))
```

bivd.object Bivariate Distribution Object

DESCRIPTION

These are S version 4 objects of class`"bivd"` .

It has child classes for each Extreme Value copula, Archimedean copula, Archimax copula (which is `bb4.copula`), `normal.copula`, and `normal.mix.copula`.

GENERATION

This class of objects is constructed from the `bivd.copula(Xmarg, Ymarg, param.Xmarg, param.Ymargin)` function.

STRUCTURE

The following slots must be present in a legitimate `"bivd.copula"` class object:

`Xmarg` : abbreviated name of the marginal distribution for `X`. If the distribution name is `dist`, it is assumed that the functions `ddist`, `pdist`, and `qdist` are implemented in Splus. For example, if `Xmarg="norm"`, functions `dnorm`, `pnorm`, and `qnorm` should be available for correct performance of the bivariate distribution functions. The distribution names are not limited to those standard for Splus, assuming that the user creates appropriate density and quantile functions.

`Ymarg` : abbreviated name of the marginal distribution for `Y`.

`param.Xmargin` : parameters for the marginal distribution of `X`. The number, and the order of the parameters should correspond to those of the functions `ddist`, `pdist`, and `qdist`.

`param.Ymargin` : parameters for the marginal distribution of `Y`.

METHODS

The `"bivd"` class of objects currently has no methods for generic functions.

Following functions are implemented for all child classes that inherit from the class `"bivd"`: `pbivd`, `dbivd`, `rbivd`, `persp.dbivd`, `persp.pbivd`, `contour.dbivd`, `contour.pbivd`.

SEE ALSO

`pbivd`,`contour.pbivd.object` .

bizTime Business Time Index

DESCRIPTION

Returns the time index vector based on the business time. This is used by the moving average operators for tick-by-tick data.

```
bizTime(pos, openTime, closeTime)
```

REQUIRED ARGUMENTS

`pos` : a `"timeDate"` vector.

`openTime` : a character string with the format `"MM:SS"` giving the business opening time.

`closeTime` : a character string with the format `"MM:SS"` giving the business closing time.

VALUE

a time index vector based on the business time in units of business days.

DETAILS

The `openTime` and `closeTime` are used to calculate the length of a business day. Based on the length of a business day, the times in `pos` are rescaled in units of business days.

To speed up the operation, it is assumed that the business days in `pos` are contiguous, meaning that there is at least one tick from each business day represented by `pos`.

SEE ALSO

`timeDate`.

EXAMPLE

```
# hourly observations from U.S. stock markets
bizTime(timeCalendar(h=10:16), "09:30", "16:00")
```

black.ts   Real Monthly Stock Return Data

SUMMARY
>   This is a monthly `"timeSeries"` object from January 1978 to December 1987, representing real monthly returns of certain assets, which is constructed from `berndt.dat`. It has the following columns:

    BOISE : real monthly returns of Boise.

 CITCRP : real monthly returns of Citicorp.

  CONED : real monthly returns of Consolidated Edison.

 CONTIL : real monthly returns of Continental Illinois.

 DATGEN : real monthly returns of Data General.

     DEC : real monthly returns of Digital Equipment Company.

  DELTA : real monthly returns of Delta Airlines.

 GENMIL : real monthly returns of General Mills.

 GERBER : real monthly returns of Gerber.

     IBM : real monthly returns of International Business Machines.

 MARKET : real value-weighted composite monthly returns based on transactions from the New York Stock Exchange and the American Exchange.

  MOBIL : real monthly returns of Mobile.

  PANAM : real monthly returns of Pan American Airways.

   PSNH : real monthly returns of Public Service of New Hampshire.

  TANDY : real monthly returns of Tandy.

 TEXACO : real monthly returns of Texaco.

  WEYER : real monthly returns of Weyerhauser.

SOURCE
>   Berndt, E. R. (1991). *The Practice of Econometrics: Classic and Contemporary*. Addison-Wesley Publishing Co.

bmw Daily log returns on BMW share price

SUMMARY

This is the daily `"timeSeries"` object for log returns on BMW share price from 1/2/1973 to 7/23/1996. Note no trading takes place at the weekend.

bond.billprice   U.S. Treasury Bill Price

DESCRIPTION
>Computes the price of U.S. treasury bills given quoted yield on a bank discount basis.

>```
bond.billprice(yield, days.to.mature, principal=1000)
```

REQUIRED ARGUMENTS

`yield` :  annualized yield on a bank discount basis.

`days.to.mature` :  the number of days to mature for the Treasury bill.

OPTIONAL ARGUMENTS

`principal` :  the principal value of the Treasury bill.

VALUE
>price of the Treasury bill.

REFERENCES
>Lynch, J. J., and Mayle, J. H. (1986). *Standard Securities Calculation Methods: Fixed Income Securities Formulas*.  New York:  Securities Industry Association.

SEE ALSO
>`bond.discount`,`bond.forward` ,`bond.spot` .

**bond.discount**   Conversion Between Discount Rate, Forward Rate and Spot Rate

DESCRIPTION

Converts between discount rate, forward rate and spot rate for bond securities.

```
bond.discount(x, maturity.in.years, input.type="spot",
              compounding.frequency=2, na.rm=F)
bond.forward(x, maturity.in.years, input.type="spot",
              compounding.frequency=2, na.rm=F)
bond.spot(x, maturity.in.years, input.type="discount",
            compounding.frequency=2, na.rm=F)
```

REQUIRED ARGUMENTS

x : a numeric vector, or a `"timeSeries"` object with a numeric vector in the data slot, which represents the rates of bond securities.

maturity.in.years : a numeric vector with length 1 or equal to the length of x, representing the maturity of the bonds in units of years.

OPTIONAL ARGUMENTS

input.type : a character string specifying the type of the input rates. Valid choices are as follows: `"spot"` and `"forward"` for `bond.discount`, `"spot"` and `"discount"` for `bond.forward`, `"discount"` and `"forward"` for `discount.spot`.

compounding.frequency : an integer specifying the compounding frequency in a year. If `compounding.frequency=0`, then continuous compounding is used. The default is 2, which corresponds to half year compounding.

na.rm : a logical flag: if `TRUE`, missing values will be removed before conversion. The default is `FALSE`.

VALUE

the discount rate for `bond.discount`, the forward rate for `bond.forward`, and the spot rate for `bond.spot`.

REFERENCES

Lynch, J. J., and Mayle, J. H. (1986). *Standard Securities Calculation Methods: Fixed Income Securities Formulas*. New York: Securities Industry Association.

SEE ALSO

`bond.billprice`, `term.struct` .

EXAMPLE

```
# convert forward rates to spot rates
bond.spot(mk.fwd2[1,1:5], mk.maturity[1:5], input="forward")
```

brazil.coffee Brazil coffee and Columbia coffee prices and their log returns.

SUMMARY

This is a `"timeSeries"` object of 2 columns data: coffee price and its log return on each of the 1518 trading days between 3/10/1993 to 1/1/1999.

BVAR.control Set Hyper-parameters for Bayesian VAR Model

DESCRIPTION

Allows users to set the hyper-parameters for Bayesian vector autoregression model.

```
BVAR.control(L0=0.9, L1=0.1, L2=1, L3=1, L4=0.05, mu5=5,
             mu6=5)
```

OPTIONAL ARGUMENTS

L0 : a positive hyper-parameter that controls the overall tightness of the prior on the error covariance matrix.

L1 : a positive number which specifies the prior standard deviation of the diagonal elements of the AR(1) coefficient matrix. It reflects how closely the random walk approximation is to be imposed. Lowering L1 toward zero has the effect of shrinking the diagonal elements of the matrix toward one and all other elements toward zero.

L2 : a hyper-parameter between zero and one, used to differentiate the lags of the dependent variables and other variables. Decreasing L2 toward zero has the effect of shrinking the off-diagonal elements of the autoregressive coefficient matrices toward zero, while setting L2 to unity means that no distinction is made between the lags of the dependent variables and other variables. Note that L2 is not used for Sims-Zha type prior.

L3 : a positive hyper-parameter that is used to determine the extent to which coefficients on lags beyond the first one are likely to be different from zero. Increasing L3 has the effect of shrinking the coefficients on high-order lags toward zero. If L3 is set to one, the rate of decay in the weight is harmonic.

L4 : a positive hyper-parameter that controls the overall tightness of the prior on the constant terms. Lowering L4 toward zero has the effect of shrinking the constant terms toward zero.

mu5 : a positive hyper-parameter that controls the unit root prior. As mu5 gets bigger, the added dummy observations will be weighted correspondingly so that the estimated VAR will increasingly satisfy the sum of coefficients restriction reflecting the unit root behavior.

mu6 : a positive hyper-parameter that controls the co-integration prior. As mu6 gets bigger, the dummy observations will be weighted correspondingly so that the estimated VAR will exhibit a co-integration relationship.

VALUE

   a list containing the values used for each of the hyper-parameters.

REFERENCES

   Robertson, J. C., and Tallman, E. W. (1999). Vector autoregressions:
   forecasting and reality. *Federal Reserve Bank of Atlanta Economic
   Review*, First Quarter, 1999.

   Sims, C. A., and Zha, T. (1998). Bayesian methods for dynamic mul-
   tivariate models. *International Economic Review*, 39(4):949-968.

EXAMPLE

```
zpolicy.mod = BVAR(cbind(CP,GDP,U)~ar(6), data=policy.dat,
                unit.root.dummy=T, coint.dummy=T,
                control=BVAR.control(L0=0.57, L1=0.13, L4=0.1))
```

BVAR.fit Fitting of Bayesian VAR Models

DESCRIPTION

>Called by `BVAR` function to fit the models. It is not supposed to be called by the users directly.

>```
BVAR.fit(X, Y, prior.mu=matrix(0, p, n), prior.Sigma=
        matrix(0, p, p))
```

REQUIRED ARGUMENTS

X, Y : numeric matrices that represent the regressors and multivariate response respectively.

OPTIONAL ARGUMENTS

prior.mu : a matrix representing the mean of the prior distribution for regression coefficients.

prior.Sigma : a matrix representing the inverse of the covariance matrix of the prior distribution for regression coefficients for each equation.

BVAR.object   Bayesian Vector Autoregression Model Objects

DESCRIPTION

These are objects of class `"BVAR"` which inherit from the class `"VAR"` and represent the fit of a Bayesian vector autoregression model.

GENERATION

This class of objects is returned from the `BVAR` function.

METHODS

The `"BVAR"` class of objects currently has methods for the following generic functions:

`coef`, `residuals`, `fitted`, `vcov`, `formula`, `plot`, `predict`, `cpredict`, `print`, `summary`.

STRUCTURE

The following components must be present in a legitimate `"BVAR"` object, in addition to components of a legitimate `"VAR"` object:

`control` : a list of hyper-parameters used for Bayesian VAR models, usually returned by a call to `BVAR.control` function.

`unit.root.dummy` : a logical flag: if `TRUE`, dummy observations were added to the sample to reflect the unit root prior.

`coint.dummy` : a logical flag: if `TRUE`, dummy observations were added to the sample to reflect the cointegration prior.

`mApprox` : a logical flag: if `TRUE`, the lag decay rate was approximated according to the harmonic decay pattern at a quarterly frequency.

SEE ALSO

`BVAR`, `VAR` , `VAR.object` .

BVAR Fit a Bayesian Vector Autoregression Model

DESCRIPTION

Returns an object of class `"BVAR"` that represents a Bayesian VAR model fit.

```
BVAR(formula, data, subset, na.rm=F, contrasts=NULL, start=NULL,
     end=NULL, demean="none", unit.root.dummy=F,
     coint.dummy=F, prior="sz", control=BVAR.control(),
     mApprox=T, ...)
```

REQUIRED ARGUMENTS

formula : a `"formula"` object, with the response on the left of a $\sim$ operator and the terms, separated by + operators, on the right. The response must be a `"matrix"` object if the data argument is not specified.

OPTIONAL ARGUMENTS

data : a data frame or `"timeSeries"` data frame in which to interpret the variables named in the `formula` and `subset` arguments. If `data` is missing, the variables in the model formula should be in the search path, and they cannot be `"timeSeries"` or `"data.frame"` objects.

subset : an expression specifying which subset of observations should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), a numeric vector indicating the observation numbers to be included, or a character vector of the observation names that should be included. All observations are included by default.

na.rm : a logical flag: if `TRUE`, missing values will be removed before fitting the model. The default is `FALSE`.

contrasts : a list giving contrasts for some or all of the factors appearing in the model formula. An element in the list should have the same name as the factor variable it encodes, and it should be either a contrast matrix (any full-rank matrix with as many rows as there are levels in the factor), or a function that computes such a matrix given the number of levels.

start : a character string which can be passed to `timeDate` function to specify the starting date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

end : a character string which can be passed to `timeDate` function to specify the ending date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

`demean` : an optional argument only used if an intercept term is not included in the model. It can be a character string or a numeric vector which specifies how to demean the multiple time series. If `demean` is a character string, there are three valid choices: `"none"`, which does not demean the time series; `"all"`, which uses the mean of all observations to demean the time series; or `"sample"`, which uses the mean of the right hand side observations to demean the time series. If `demean` is a numeric vector, it must have the same length as the dimension of the multiple time series, and will be used to demean the time series. The default is `"none"`.

`unit.root.dummy` : a logical flag: if `TRUE`, some dummy observations will be added to the sample which reflects unit root prior. The default `FALSE`.

`coint.dummy` : a logical flag: if `TRUE`, some dummy observations will be added to the sample which reflects co-integration prior. The default `FALSE`.

`prior` : a character string specifying the type of prior distributions for the model parameters. Currently the only choice is `"sz"`, which uses the Sims-Zha type prior distribution.

`control` : a list of hyper-parameters which determines the tightness of prior distributions, usually returned by a call to the function `BVAR.control`. See `BVAR.control` for their names and default values.

`mApprox` : a logical flag: if `TRUE`, the lag decay rate will be approximated according to the harmonic decay pattern at a quarterly frequency. This is usually used for monthly data. The default is `TRUE`.

`...` : any optional arguments that can be passed to the `timeDate` function to parse the `start` or `end` specification.

VALUE

an object of class `"BVAR"` representing the fit of the Bayesian VAR model. See `BVAR.object` for details.

REFERENCES

Kadiyala, K. R., and Karlsson, S. (1993). Forecasting with generalized Bayesian vector autoregressions. *Journal of Forecasting*, 12:365-378.

Robertson, J. C., and Tallman, E. W. (1999). Vector autoregressions: forecasting and reality. *Federal Reserve Bank of Atlanta Economic Review*, First Quarter, 1999.

Sims, C. A., and Zha, T. (1998). Bayesian methods for dynamic multivariate models. *International Economic Review*, 39(4):949-968.

SEE ALSO

`BVAR.object`, `VAR` .

EXAMPLE

```
# use a time series data frame
pol.mod = BVAR(cbind(CP,GDP,U)~ar(6), data=policy.dat)

# use a matrix directly without the data argument
pol.mat = as.matrix(seriesData(policy.dat))
pol.mod = BVAR(pol.mat~ar(6), unit.root.dummy=T, coint.dummy=T)
```

CheckSsf Enforce a State Space Form

DESCRIPTION

> Returns an object representing a state space form given minimal necessary information.

>   CheckSsf(ssf)

REQUIRED ARGUMENTS

> ssf : a list which contains necessary components for a valid state space form. It must contain at least the `mPhi` and `mOmega` components. See the returned object for details.

VALUE

> an S version 3 object of class `"ssf"` which represents the full state space form of the input list `ssf`. It must contain the following components:

> cSt: an integer that specifies the number of state variables in the state space form.

> cY: an integer that specifies the number of response variables, or observables, in the state space form.

> cX: an integer that specifies the number of time varying variables in the state space form.

> cT: an integer that specifies the number of observations for the time varying variables if any.

> mX: a `cT x cX` matrix that gives the values of the time varying variables.

> mPhi: a `(cSt + cY) x cSt` matrix representing the coefficient matrix of the state variables. Note that the top `cSt` rows correspond to the state equation or transition equation, while the last `cY` rows correspond to the measurement equation.

> mJPhi: a `(cSt + cY) x cSt` matrix that determines the time varying aspect of `mPhi`. If any element of `mJPhi` is positive, then the corresponding element in `mPhi` is time varying, and the time varying coefficients are given in the `mX` component. For example, if the `(1,1)` element of `mJPhi` is 2, then the time varying coefficients for the `(1,1)` element of `mPhi` is given in the second column of `mX`. If there is no time varying coefficient in `mPhi` then `mJPhi` is simply zero.

`mOmega:` a `(cSt + cY) x (cSt + cY)` matrix representing the variance-covariance matrix of disturbance terms. Note that the first `cSt x cSt` block gives the covariance matrix for the state disturbance, while the last `cY x cY` block gives the covariance matrix for the response disturbance.

`mJOmega:` a `(cSt + cY) x (cSt + cY)` matrix that determines the time varying aspect of `mOmega`. If any element of `mJOmega` is positive, then the corresponding element in `mOmega` is time varying, and the time varying coefficients are given in the `mX` component. For example, if the `(1,1)` element of `mJOmega` is 3, then the time varying coefficients for the `(1,1)` element of `mOmega` is given in the third column of `mX`. If there is no time varying coefficient in `mOmega` then `mJOmega` is simply zero.

`mDelta:` a `(cSt + cY) x 1` matrix representing the constant terms in the state space model. Note that the first `cSt` elements represent the constant term in the state equations or transition equations, and the last `cY` elements represent the constant term in the measurement equation.

`mJDelta:` a `(cSt + cY) x 1` matrix that determines the time varying aspect of `mDelta`. If any element of `mJDelta` is positive, then the corresponding element in `mDelta` is time varying, and the time varying coefficients are given in the `mX` component. For example, if the `(1,1)` element of `mJDelta` is 3, then the time varying coefficients for the `(1,1)` element of `mDelta` is given in the third column of `mX`. If there is no time varying coefficient in `mDelta` then `mJDelta` is simply zero.

`mSigma:` a `(cSt + 1) x cSt` matrix with the first `cSt` rows giving the initial covariance matrix of the state variables, and the last row giving the initial mean of the state variables.

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Koopman, S. J., Shephard, N., and Doornik, J. A. (1999). Statistical algorithms for methods in state space form using SsfPack 2.2. *Econometric Journal*, 2:113-166.

EXAMPLE

```
ar.mod = GetSsfArma(ar=c(0.5, 0.2))
CheckSsf(ar.mod)
```

coef.arima.rob Use `coef()` on an `arima.rob` Object

DESCRIPTION

This is a method for the function `coef()` for objects inheriting from class `"arima.rob"`. See `coef` or `coef.default` for the general behavior of this function and for the interpretation of object.

```
coef.arima.rob(object)
```

coef.FARIMA   Use `coef()` on a `FARIMA` Object

> `coef.FARIMA(object, ...)`
>
> This is a method for the function `coef()` for objects inheriting from class `"FARIMA"`. See `coef` or `coef.default` for the general behavior of this function and for the interpretation of `object`.

coef.mgarch Extracts GARCH Model Coefficients

DESCRIPTION
>   Extracts coefficients from an object of class `mgarch`. The function
>   `coef.mgarch` is usually invoked by calling the generic function `coef`.

>   `coef(x)`

REQUIRED ARGUMENTS

>   `x` : an object of class `"mgarch"`.

 VALUE
>   coefficients of the GARCH model.

SEE ALSO
>   `garch`, `mgarch` , `summary.mgarch` , `print.mgarch` , `print.garch` .

EXAMPLE

>   ```
>   hp.s.mod = garch(hp.s~1, ~garch(1,1))
>   coef(hp.s.mod)
>   ```

coef.mOLS Use `coef()` on an `mOLS` Object

> `coef.mOLS(object, ...)`
>
> This is a method for the function `coef()` for objects inheriting from class `"mOLS"`. See `coef` or `coef.default` for the general behavior of this function and for the interpretation of `object`.

coef.OLS Use `coef()` on an `OLS` Object

> `coef.OLS(object, ...)`
>
> This is a method for the function `coef()` for objects inheriting from class `"OLS"`. See `coef` or `coef.default` for the general behavior of this function and for the interpretation of `object`.

coef.RLS Use `coef()` on an RLS Object

> `coef.RLS(object, ...)`
>
> This is a method for the function `coef()` for objects inheriting from class `"RLS"`. See `coef` or `coef.default` for the general behavior of this function and for the interpretation of `object`.

coef.rollOLS Use `coef()` on a `rollOLS` Object

`coef.rollOLS(object, ...)`

This is a method for the function `coef()` for objects inheriting from class `"rollOLS"`. See `coef` or `coef.default` for the general behavior of this function and for the interpretation of `object`.

coef.SEMIFAR Use `coef()` on a SEMIFAR Object

> `coef.SEMIFAR(object, ...)`
>
> This is a method for the function `coef()` for objects inheriting from class `"SEMIFAR"`. See `coef` or `coef.default` for the general behavior of this function and for the interpretation of `object`.

coef.SUR   Use `coef()` on a SUR Object

> `coef.SUR(object, ...)`
>
> This is a method for the function `coef()` for objects inheriting from class `"SUR"`. See `coef` or `coef.default` for the general behavior of this function and for the interpretation of `object`.

coef.unitroot   Use coef() on a unitroot Object

> coef.unitroot(object, ...)
>
> This is a method for the function coef() for objects inheriting from class "unitroot". See coef or coef.default for the general behavior of this function and for the interpretation of object.

coef.VAR Use `coef()` on a `VAR` Object

> `coef.VAR(object, ...)`
>
> This is a method for the function `coef()` for objects inheriting from class `"VAR"`. See `coef` or `coef.default` for the general behavior of this function and for the interpretation of `object`.

coint.object Johansen Cointegration Test Objects

DESCRIPTION

These are objects of class `"coint"` which represent Johansen rank tests for cointegration.

GENERATION

This class of objects is returned from the `coint` function.

METHODS

The `"coint"` class of objects currently has methods for the following generic functions:

`print`, `summary`.

STRUCTURE

The following components must be present in a legitimate `"coint"` object:

`call` : an image of the call that produced the object.

`evals` : a numeric vector with length `k + c` representing the eigenvalues, where `k` is the number of dependent variables, `c` is 1 if `trend` is `"rc"` or `"rt"` and 0 otherwise.

`coint.vectors` : a numeric matrix of dimension `(k+c) x (k+c)`, with the first `k x k` sub-matrix representing the normalized cointegrating vectors in rows, where `k` is the number of dependent variables, `c` is 1 if `trend` is `"rc"` or `"rt"` and 0 otherwise.

`adj.coef` : a numeric matrix of dimension `k x (k+c)`, with the first `k x k` sub-matrix representing adjustment coefficients in rows, where `k` is the number of dependent variables, `c` is 1 if `trend` is `"rc"` or `"rt"` and 0 otherwise.

`loglike.values` : a numeric vector with length `k + c`, with the first `k` values representing the log-likelihood values, where `k` is the number of dependent variables, `c` is 1 if `trend` is `"rc"` or `"rt"` and 0 otherwise.

`maxeig.tests` : a numeric vector with length `k + c`, with the first `k` values representing the maximum eigenvalue tests, where `k` is the number of dependent variables, `c` is 1 if `trend` is `"rc"` or `"rt"` and 0 otherwise.

`trace.tests` : a numeric vector with length `k + c`, with the first `k` values representing the trace tests, where `k` is the number of dependent variables, `c` is 1 if `trend` is `"rc"` or `"rt"` and 0 otherwise.

`trend` : a character string giving the trend specification in the original call which generated the object. See the help file for `coint` for details.

ar.order : an integer giving the autoregressive order of the VECM. Note that this is one less than the autoregressive order of the corresponding VAR representation.

nobs : an integer giving the number of observations used in the estimation.

df : an integer giving the degree of freedom of residuals.

S00, S01, S11 : the residual moment matrices of the Johansen regressions.

Z0, Z1, Z2 : the matrices giving the VECM representation. This is only present if save.VECM=T in the original call which generated the object.

SEE ALSO

coint.

coint   Johansen Rank Tests for Cointegration

DESCRIPTION

> Returns Johansen rank tests for cointegration based on maximum likelihood estimation of the vector error correction model (VECM).

```
coint(Y, X=NULL, lags=1, trend="c", save.VECM=T)
```

REQUIRED ARGUMENTS

> **Y** : a matrix, or a data frame, or a `"timeSeries"` object with a matrix or a data frame in the data slot. This represents the multivariate time series each of which is usually assumed to be integrated of order 1.

OPTIONAL ARGUMENTS

> **X** : a numeric object representing the exogenous variables in the model if present. Note that `X` must have the same number of observations as `Y`, or must have the same number of observations after adjusting for lags in `Y`. It should not contain a constant term, or a linear time trend.

> **lags** : an integer specifying the number of lags or autoregressive terms to use in VECM. Note that this is one less than the AR order of the corresponding VAR representation. The default is 1, which corresponds to a VAR(2) representation.

> **trend** : a character string specifying the trend assumption of VECM. Valid choices are: `"nc"` for no constant, `"c"` for unrestricted constant, `"rc"` for restricted constant, `"ct"` for unrestricted trend, and `"rt"` for restricted trend. The default is `"c"`.

> **save.VECM** : a logical flag: if `TRUE`, the VECM representation is saved in the returned object. This is necessary if a further VECM estimation is desired. The default is `TRUE`.

VALUE

> an object of class `"coint"` representing Johansen's rank tests for cointegration. See `coint.object` for details.

DETAILS

> The critical values of Johansen's tests are currently only available if the system has less than twelve variables. If the system involves twelve or more variables, the critical values are not printed.

REFERENCES

> Johansen, S. (1995). *Likelihood-based Inference in Cointegrated Vector Autoregressive Models*. Oxford University Press.

SEE ALSO

VECM,cancor ,coint.object ,stationaryTest ,unitroot .

EXAMPLE

```
coint(mk.zero2[,1:5], lags=1, trend="rc", save=F)
```

collinearTest.OLS Use `collinearTest()` on an `OLS` Object

DESCRIPTION
>This is a method for the function `collinearTest()` for objects inheriting from class `"OLS"`. See `collinearTest` for the general behavior of this function.

>`collinearTest.OLS(x, method="cn")`

REQUIRED ARGUMENTS

>`x` : an object of class `"OLS"`. This is usually returned by a call to `OLS` function. regression.

OPTIONAL ARGUMENTS

>`method` : a character string specifying the type of test to compute. Valid choices are: `"cn"` for condition number, and `"vif"` for variance inflation factor. The default is `"cn"`.

SEE ALSO
>`collinearTest`.

collinearTest Test for Multi-Collinearity in a Regressor Matrix

DESCRIPTION

Returns the selected test statistic for multi-collinearity in a regressor matrix.

This function is an S Version 3 generic (see `Methods`); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: `OLS`, `lm`.

```
collinearTest(x, method="cn")
```

REQUIRED ARGUMENTS

x : a matrix, or a `"timeSeries"` object with a matrix in the data slot, which represents the regressor matrix for a regression.

OPTIONAL ARGUMENTS

method : a character string specifying the type of test to compute. Valid choices are: `"cn"` for condition number, and `"vif"` for variance inflation factor. The default is `"cn"`.

VALUE

the condition number if `method="cn"`, or a numeric vector representing the variance inflation factor. `x`.

DETAILS

A condition number larger than 20 is usually a sign of multi-collinearity. If any variance inflation factor exceeds 10, it also indicates multi-collinearity.

REFERENCES

Belsley, D. A., Kuh, E., and Welsch, R. E. (1980). *Regression Diagnostics: Identifying Influential Data And Sources Of collinearity*. New York: Wiley.

Greene, W. H. (2000). *Econometric Analysis*. Hemel Hempstead: Prentice Hall.

SEE ALSO

`collinearTest.OLS`,`heteroTest` .

colSkewness   Column Summaries

DESCRIPTION

Sample skewness, sample kurtosis, and cumulative sums by column.

```
colSkewness(x, na.rm=F)
colKurtosis(x, na.rm=F)
colCumsums(x, na.rm=F)
```

REQUIRED ARGUMENTS

x : a vector, matrix, data frame, or a `"timeSeries"` object with a numeric
   object in the data slot.

OPTIONAL ARGUMENTS

na.rm : a logical flag: if `TRUE`, missing values will be removed before the com-
   putation. The default is `FALSE`.

VALUE

`colSkewness` returns the sample skewness by column, `colKurtosis`
returns the sample kurtosis by column, and `colCumsums` returns the
cumulative sums by column.

SEE ALSO

`colMeans`,`cummax` ,`cumsum` ,`skewness` .

compare.mgarch GARCH Model Comparison

DESCRIPTION

Computes model comparison statistics.

```
compare.mgarch(...)
```

REQUIRED ARGUMENTS

...: a number of fitted objects of class `"mgarch"` and `"garch"` to be compared.

VALUE

a list containing model comparison statistics including AIC and BIC.

SEE ALSO

`plot.compare.garch`, `plot.compare.mgarch`, `print.compare.mgarch`
.

contour.pbivd 3D Surface and Contour Plots of Bivariate CDF and PDF

DESCRIPTION

Creates a 3D surface or contour plot of bivariate cumulative distribution function and density.

```
persp.dbivd(dist, n=50, xlim=NA, ylim=NA, ...)
persp.pbivd(dist, n=50, xlim=NA, ylim=NA, ...)
contour.dbivd(dist, n=50, xlim=NA, ylim=NA, ...)
contour.pbivd(dist, n=50, xlim=NA, ylim=NA, ...)
```

REQUIRED ARGUMENTS

dist : an object of class `"bivd"`.

OPTIONAL ARGUMENTS

n : specifies the number of grid points along each axis used for the evaluation of `pdivd` or `dbivd` functions. For example, if n=50 (default), `pcopula` is evaluated at 50*50 = 2500 points. Smaller values will produce a lower resolution plots, higher values higher resolution.

... : additional parameters that will be passed to the S-Plus function `guiPlot` that is used to create a 3D surface or countour plot (see S-Plus help file for `guiPlot` for the list of available parameters). One of such additional parameters is `GraphSheet`, which allows the new graph to be added to one of the existing plots, instead of creating a new graph sheet (default).

VALUE

a list with components x, y, and z. The components x and y are vectors, z is a matrix with dimensions `c(length(x), length(y))`, and `z[i,j] = pbivd(copula,x[i], y[j])` or `z[i,j] = dbivd(copula,x[i], y[j])` for functions `*.pbivd` and `*.dbivd`, respectively. A 3D surface or a contour plot is produced.

SEE ALSO

`contour.pcopula`.

EXAMPLE

```
bivd1 = bivd(normal.copula(0.6), "norm", "t",
    param.Xmargin=(0, 2.5), param.Ymargin=5)
persp.dbivd(bivd1)
```

contour.plot 3D Surface and Contour Plots of Copula CDF and PDF

DESCRIPTION

Creates a 3D surface or contour plot of copula CDF and density.

```
contour.plot(object, n=100, nlevels=10, add=F, ...)
persp.dcopula(copula, n=20, ...)
persp.pcopula(copula, n=20, ...)
contour.dcopula(copula, n=50, ...)
contour.pcopula(copula, n=50, ...)
```

REQUIRED ARGUMENTS

`copula` : an object of class `"copula"`.

OPTIONAL ARGUMENTS

`n` : specifies the number of grid points along each axis used for the evaluation of `pcopula` or `dcopula` function. For example, if `n=20` (default) `pcopula` is evaluated at `20^2 = 40` points. Smaller values will produce a lower resolution plots, higher values higher resolution.

`...` : additional parameters that will be passed to the S-Plus function `guiPlot` that is used to create a 3D surface or contour plot (see S-Plus help file for `guiPlot` for the list of available parameters). One of such additional parameters is `GraphSheet`, which allows the new graph to be added to one of the existing plots, instead of creating a new graph sheet (default).

`nlevels` : the number of contour lines to draw on the screen for a plot produced by the function `contour.plot`.

VALUE

a list with components x, y, and z. The components x and y are vectors, z is a matrix with dimensions `c((length(x), length(y))`, and `z[i,j] = pcopula(copula, x[i], y[j])` or `z[i,j] = dcopula(copula, x[i], y[j])` for functions `*.pcopula` and `*.dcopula`, respectively. A 3D surface or a contour plot is produced.

DETAILS

The generic function `contour.plot` is similar to the function`contour.pcopula` , only instead of producing a contour plot using `guiPlot("Contour", ...)` function, it uses`contour` function. In the latter case, additional arguments that correspond to graphical parameters (such as `col`) can be included (see S-Plus help files for functions`contour` and `par`).

If `copula` is an instance of `empirical.copula` class, the copula density is computed using 2D kernel density estimator implemented by Venables and Ripley (1997) (function `kde2d` in library MASS, see Venables and Ripley (1997) and S-Plus help files for details). You might need to add the MASS library to your Splus search path before using this function.

SEE ALSO

`contour.pbivd`.

EXAMPLE

```
# Please close all of the Splus graphics windows
cop2 <- normal.copula(0.6)
contour.dcopula(cop2)
contour.dcopula(normal.copula(0.7), GraphSheet="GS1")
ttt <- empirical.copula(rcopula(cop2,20000))
persp.dcopula(ttt, n = 30)
persp.dcopula(cop2, n = 30)
persp.pcopula(ttt, n = 30)
persp.pcopula(cop2, n = 30)
# Please close all of the Splus graphics windows
contour.pcopula(ttt, n = 30)
contour.dcopula(cop2, n = 30, GraphSheet = "GS1")
```

copula.object   Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"copula"` .

Copula class has child classes `ev.copula`, `archm.copula`, and `archimax.copula` (which is `bb4.copula`). Child class `ev.copula` represents Extreme Value copulas, class `archm.copula` represents Archimedean copulas, and `archimax.copula` represents Archimax copulas. We refer to these classes as copula types.

Each parametric family of copulas that belongs to one of theses three copula types is defined as a separate sub-class, and is named `<<family.name>>.copula` (e.g. `normal.copula`). If the family defines an Extreme Value copula, it's considered to be of `ev.copula` type. If it is an Archimedean, but not Extreme Value copula, it inherits from `archm.copula` class (some copula families are both Extreme Value and Archimedean copulas at the same time, e.g. the Gumbel copulas). Otherwise, if the family can be represented as an Archimax copula, it is considered to be of `archimax.copula` type.

A copula's type will determine what functions are used to calculate the density, cumulative probability, generate random variable pairs and to compute Kendall's tau, Spearman's rho and the tail dependence index. All parametric families of copula inherit from this fundamental common class `copula`.

For each parametric copula of class `archm.copula`, the Archimedean generator `PHI(copula,t)`, its derivative `PhiDer(copula,t)`, inverse `InvPhi(copula,t)`, and the derivatives of the inverse `InvPhifirstDer(copula,t)` and `InvPhisecondDer(copula,t)` are defined.

For each parametric copula of class `ev.copula`, the dependence function for an extreme value copula `Afunc(copula,t)`, its first derivatives `AfirstDer(copula,t)`, the second derivatives `AsecondDer(copula,t)`, and `Hderiv(copula,t)` are defined.

STRUCTURE

The following slots must be present in a legitimate `"archm.copula"` class object:

`parameters` : a vector with values of the parameters for this particular instance of the object `copula` (the length of this vector is equal to the number of parameters allowed for this particular family).

`param.names` : typical greek letters that are used for the parameters (such as `delta` or `theta`).

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s).

param.upbnd : similar to **param.lowbnd**, only values of the upper bounds for the parameter(s).

message: a message specifying the name of the parametric copula family, and the copula sub-class, such as Archimedean copula, if applicable.

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

ev.copula.object,archm.copula.object .

CPI.dat Seasonally Adjusted U.S. Consumer Price Index (CPI)

SUMMARY

This is a monthly **"timeSeries"** object from January 1913 to November 2001, representing seasonally adjusted U.S. Consumer Price Index (CPI).

cpredict.BVAR   Use `cpredict()` on a `BVAR` Object

DESCRIPTION

     Performs prediction from a fitted Bayesian VAR model, conditional on some knowledge of future path of the response variables.

```
cpredict.BVAR(object, n.predict=1, newdata=NULL, olddata=NULL,
  variables.conditioned=NULL, steps.conditioned=NULL,
  upper=NULL, lower=NULL, middle=NULL, seed=100, n.sim=1000)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"BVAR"`.

OPTIONAL ARGUMENTS

`n.predict` : an integer specifying the number of periods to predict ahead.

`newdata` : a data frame containing the future values of exogenous variables, if any, at which predictions are required.

`olddata` : a data frame containing the original data used to fit `object`. This is only required if `tslag` is used to create distributed lags of exogenous variables in the original call that generated `object`.

`variables.conditioned` : a numeric or character vector specifying the variables in the responses for which conditions will be imposed. If it is a numeric vector, each element of the vector represents the order of the variable in the multivariate response; if it is a character vector, each element of the vector represents the name of the variable in the multivariate response. If `NULL`, conditions will be imposed on all the variables.

`steps.conditioned` : a numeric vector specifying upon which forecasts conditions will be imposed. If `NULL`, conditions will be imposed on all the steps smaller than or equal to `n.period`.

`upper` : a numeric matrix specifying the upper limit of future response, with each row corresponds to steps in the future, and each column corresponds to variables. This can be used (with `lower`) to impose soft conditions. If `NULL` and `lower` is given, the upper limit is taken to be infinity.

`lower` : a numeric matrix specifying the lower limit of future response, with each row corresponds to steps in the future, and each column corresponds to variables. This can be used (with `upper`) to impose soft conditions. If `NULL` and `upper` is given, the lower limit is taken to be negative infinity.

`middle` : a numeric matrix specifying hard conditions on future response, with each row corresponds to steps in the future, and each column corresponds to variables.

`seed` : an integer between 0 and 1023, or a previous value of `.Random.seed`, which specifies the random seed to be used.

`n.sim` : an integer specifying the number of simulations.

VALUE

an object of class `"forecast"`. The `draws` element of the object contains all the simulated draws that satisfied the conditions. See `forecast.object` for details. Sets the value of the `.Random.seed` object in the working directory.

DETAILS

This function is a method for the generic function `cpredict` for class `"BVAR"`. It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.BVAR` regardless of the class of the object.

REFERENCES

Waggoner, D. F., and Zha, T. (1999). Conditional forecasts in dynamic multivariate models. *Review of Economics and Statistics*, 81(4):639-651.

SEE ALSO

`cpredict`, `cpredict.VAR` , `forecast.object` , `predict` .

EXAMPLE

```
# fit a Bayesian VAR model zpolicy.mod
zpolicy.mat = as.matrix(seriesData(policy.dat[1:264,]))
zpolicy.mod = BVAR(zpolicy.mat~ar(13), unit.root.dummy=T, coint.dummy=T)
# fix the next period value for CP and FFR then predict ahead
hard.cond = matrix(c(4.6559, 0.1908), nrow=1)
zpolicy.pred = cpredict(zpolicy.mod, 12, steps=1,
                        variables=c("CP", "FFR"), middle=hard.cond)
```

cpredict Conditional Predictions/Forecasts

DESCRIPTION

Performs conditional predictions/forecasts from a fitted multivariate model object.

This function is an S Version 3 generic (see `Methods`); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: `VAR`, `BVAR`, `VECM`.

```
cpredict(object, ...)
```

REQUIRED ARGUMENTS

`object` : any object representing a fitted multivariate model.

VALUE

an object of class `"forecast"`. See `forecast.object` for details.

cpredict.VAR Use `cpredict()` on a `VAR` Object

DESCRIPTION

Performs prediction from a fitted VAR model, conditional on some knowledge of future path of the response variables.

```
cpredict.VAR(object, n.predict=1, newdata=NULL, olddata=NULL,
  method="mc", unbiased=T, variables.conditioned=NULL,
  steps.conditioned=NULL, upper=NULL, lower=NULL, middle=NULL,
  seed=100, n.sim=1000)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"VAR"`.

OPTIONAL ARGUMENTS

`n.predict` : an integer specifying the number of periods to predict ahead.

`newdata` : a data frame containing the future values of exogenous variables, if any, at which predictions are required.

`olddata` : a data frame containing the original data used to fit `object`. This is only required if `tslag` is used to create distributed lags of exogenous variables in the original call that generated `object`.

`method` : a character string specifying the prediction method. This is only used if soft conditions are imposed. Valid choices are: `"mc"`, for which the conditional forecasts are computed using Monte Carlo simulation; and `"bootstrap"`, for which the conditional forecasts are computed using bootstrap. The default is `"mc"` for conditional forecasts.

`unbiased` : a logical flag; if `TRUE`, unbiased estimate of error covariance will be used; if `FALSE`, the maximum likelihood estimate of error covariance will be used. The default is `TRUE`.

`variables.conditioned` : a numeric or character vector specifying the variables in the responses for which conditions will be imposed. If it is a numeric vector, each element of the vector represents the order of the variable in the multivariate response; if it is a character vector, each element of the vector represents the name of the variable in the multivariate response. If `NULL`, conditions will be imposed on all the variables.

`steps.conditioned` : a numeric vector specifying upon which forecasts conditions will be imposed. If `NULL`, conditions will be imposed on all the steps smaller than or equal to `n.period`.

`upper :` a numeric matrix specifying the upper limit of future response, with each row corresponds to steps in the future, and each column corresponds to variables. This can be used (with `lower`) to impose soft conditions. If `NULL` and `lower` is given, the upper limit is taken to be infinity.

`lower :` a numeric matrix specifying the lower limit of future response, with each row corresponds to steps in the future, and each column corresponds to variables. This can be used (with `upper`) to impose soft conditions. If `NULL` and `upper` is given, the lower limit is taken to be negative infinity.

`middle :` a numeric matrix specifying hard conditions on future response, with each row corresponds to steps in the future, and each column corresponds to variables.

`seed :` an integer between 0 and 1023, or a previous value of `.Random.seed`, which represents the random seed to be used. This is only used for soft conditions.

`n.sim :` an integer specifying the number of simulations. This is only used for soft conditions.

VALUE

an object of class `"forecast"`. When soft conditions are used, the `draws` element contains all the simulated draws that satisfied the soft conditions. See `forecast.object` for details. Sets the value of the `.Random.seed` object in the working directory if soft conditions are used.

DETAILS

This function is a method for the generic function `cpredict` for class `"VAR"`. It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.VAR` regardless of the class of the object.

REFERENCES

Waggoner, D. F., and Zha, T. (1999). Conditional forecasts in dynamic multivariate models. *Review of Economics and Statistics*, 81(4):639-651.

SEE ALSO

`cpredict`, `cpredict.BVAR`, `cpredict.VECM`, `forecast.object`, `predict`.

EXAMPLE

```
# fit a classical VAR model zpolicy.mod
zpolicy.mat = as.matrix(seriesData(policy.dat[1:264,]))
zpolicy.mod = VAR(zpolicy.mat~ar(13))
```

```
# fix the next period value for CP and FFR then predict ahead
hard.cond = matrix(c(4.6559, 0.1908), nrow=1)
zpolicy.pred = cpredict(zpolicy.mod, 12, steps=1,
                        variables=c("CP", "FFR"), middle=hard.cond)
```

cpredict.VECM Use `cpredict()` on a `VECM` Object

DESCRIPTION

Performs prediction from a fitted VECM, conditional on some knowledge of future path of the response variables.

```
cpredict.VECM(object, n.predict=1, newdata=NULL, method="mc",
  unbiased=T, variables.conditioned=NULL,
  steps.conditioned=NULL, upper=NULL, lower=NULL,
  middle=NULL, seed=100, n.sim=1000)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"VECM"`.

OPTIONAL ARGUMENTS

`n.predict` : an integer specifying the number of periods to predict ahead.

`newdata` : a data frame containing the future values of exogenous variables, if any, at which predictions are required.

`method` : a character string specifying the prediction method. This is only used if soft conditions are imposed. Valid choices are: `"mc"`, for which the conditional forecasts are computed using Monte Carlo simulation; and `"bootstrap"`, for which the conditional forecasts are computed using bootstrap. The default is `"mc"` for conditional forecasts.

`unbiased` : a logical flag; if `TRUE`, unbiased estimate of error covariance will be used; if `FALSE`, the maximum likelihood estimate of error covariance will be used. The default is `TRUE`.

`variables.conditioned` : a numeric or character vector specifying the variables in the responses for which conditions will be imposed. If it is a numeric vector, each element of the vector represents the order of the variable in the multivariate response; if it is a character vector, each element of the vector represents the name of the variable in the multivariate response. If `NULL`, conditions will be imposed on all the variables.

`steps.conditioned` : a numeric vector specifying upon which forecasts conditions will be imposed. If `NULL`, conditions will be imposed on all the steps smaller than or equal to `n.period`.

`upper` : a numeric matrix specifying the upper limit of future response, with each row corresponds to steps in the future, and each column corresponds to variables. This can be used (with `lower`) to impose soft conditions. If `NULL` and `lower` is given, the upper limit is taken to be infinity.

lower : a numeric matrix specifying the lower limit of future response, with each row corresponds to steps in the future, and each column corresponds to variables. This can be used (with `upper`) to impose soft conditions. If `NULL` and `upper` is given, the lower limit is taken to be negative infinity.

middle : a numeric matrix specifying hard conditions on future response, with each row corresponds to steps in the future, and each column corresponds to variables.

seed : an integer between 0 and 1023, or a previous value of `.Random.seed`, which represents the random seed to be used. This is only used for soft conditions.

n.sim : an integer specifying the number of simulations. This is only used for soft conditions.

VALUE

an object of class `"forecast"`. When soft conditions are used, the `draws` element contains all the simulated draws that satisfied the soft conditions. See `forecast.object` for details. Sets the value of the `.Random.seed` object in the working directory if soft conditions are used.

DETAILS

This function is a method for the generic function `cpredict` for class `"VECM"`. It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.VECM` regardless of the class of the object.

REFERENCES

Waggoner, D. F., and Zha, T. (1999). Conditional forecasts in dynamic multivariate models. *Review of Economics and Statistics*, 81(4):639-651.

SEE ALSO

`cpredict`, `cpredict.BVAR`, `cpredict.VAR`, `forecast.object`, `predict`
.

cspline   Cubic Spline Interpolation

DESCRIPTION
    Returns interpolations through data points using cubic spline.

    `cspline(x, y, xx)`

REQUIRED ARGUMENTS

    x : a vector, or a one-dimensional `"timeSeries"` object.

    y : a vector, or a matrix, or a `"timeSeries"` object, with the same number
        of rows in x.

    xx : a vector or a one-dimensional `"timeSeries"` object that specifies the
        x-values for which the corresponding y-values will be interpolated.

 VALUE
    a vector, or a matrix, or a `"timeSeries"` object that corresponds to
    xx.

DETAILS
    If y is a one-dimensional object, the returned value is also one-dimensional;
    if y is a two-dimensional object, the returned value is also two-dimensional
    with each column being the interpolations for each column in y. The
    not-a-knot end conditions are used for cubic spline interpolation.

REFERENCES
    de Boor, C. (1978). *A Practical Guide to Splines*. Springer-Verlag.

SEE ALSO
    `disaggregate`, `interpNA` , `spline` .

cusumTest   CUSUM Test of Structural Stability

DESCRIPTION
>   Returns the CUSUM/CUSUMSQ test statistics for structural stability.

>   `cusumTest(object, squares=T)`

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"RLS"`. This is usually returned by a call to the `RLS` function.

OPTIONAL ARGUMENTS

`squares` : a logical flag: if `TRUE`, the CUSUMSQ test will also be returned together with the CUSUM test. The default is `TRUE`.

VALUE
>   a one-dimensional object representing the CUSUM statistics if `squares=F`, or a two-dimensional object with the first column being the CUSUM statistics and the second column the CUSUMSQ statistics if `squares=T`.

REFERENCES
>   Brown, R. L., Durbin, J., and Evans, J. M. (1975). Techniques for testing the constancy of regression relationships over time. *Journal of the Royal Statistical Society B*, 37:141-192.

SEE ALSO
>   `RLS`,`plot.RLS` .

EXAMPLE
```
tmp.dat = data.frame(Loss=stack.loss, stack.x)
tmp.rls = RLS(Loss~Air.Flow+Water.Temp+Acid.Conc., data=tmp.dat)
cusumTest(tmp.rls)
```

danish Danish Fire Insurance Claims

SUMMARY

This is a `"timeSeries"` object of large fire insurance claims in Denmark from 1/3/1980 to 12/31/1990.

SOURCE

Supplied by Mette Rytgaard of Copenhagen Re.

days.count Number of Elapsed Days

DESCRIPTION

Returns the number of elapsed days between two dates according to different business conventions.

```
days.count(day1, day2, actual.days.in.month=T, days.in.year=365,
           years=F, ...)
```

REQUIRED ARGUMENTS

`day1, day2` : a `"timeDate"` vector, or a character vector which can be passed to `timeDate` function to obtain the times/dates information.

OPTIONAL ARGUMENTS

`actual.days.in.month` : a logical flag: if `TRUE`, the actual number of days in a month is used; if `FALSE`, 30 days are used for each month. The default is `TRUE`.

`days.in.year` : an integer specifying the number of days in a year. The usual choices are 365 and 360. The default is 365.

`years` : a logical flag: if `TRUE`, the results are expressed in terms of years. The default is `FALSE`.

`...` : any optional argument that may be passed to `timeDate` function to parse the times/dates information.

VALUE

a numeric vector representing the number of elapsed days between `day1` and `day2`.

SEE ALSO

`timeDate`.

EXAMPLE

```
# use actual days in a month
days.count(timeCalendar(m=2, d=1:28, y=2000),
           timeCalendar(m=2, d=1:28, y=2001))

# use 30 days in a month
days.count(timeCalendar(m=2, d=1:28, y=2000),
           timeCalendar(m=2, d=1:28, y=2001),
           actual.days.in.month=F)
```

ddexp Double Exponential Distribution

DESCRIPTION

Generates densities, cumulative probabilities, quantiles, and random numbers for the double exponential distribution.

```
ddexp(x, rate=sqrt(2))
pdexp(q, rate=sqrt(2))
qdexp(p, rate=sqrt(2))
rdexp(n, rate=sqrt(2))
```

REQUIRED ARGUMENTS

x: vector of quantiles. Missing values are allowed.

q: vector of quantiles. Missing values are allowed.

p : vector of probabilities. Missing values are allowed.

n: sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.

rate: the inverse of the mean of the distribution.

VALUE

densities (`ddexp`), probabilities (`pdexp`), quantiles (`qdexp`), or random values are returned.

The function `rdexp` creates the dataset `.Random.seed` if it does not already exist, otherwise its value is updated.

DETAILS

Elements of `p` or `q` that are missing will cause the corresponding elements of the results to be missing.

decluster Decluster Point Process

DESCRIPTION
>   Declusters the clustered point process data so that Poisson assumption
>   is more tenable over a high threshold.

>   `decluster(series, gap=NA, plot=T)`

REQUIRED ARGUMENTS

`series` : a `"timeSeries"` or `"signalSeries"` data on threshold exceedances.

`gap:` parameter to be used in the runs method; any two consecutive thresh-
old exceedances separated by more than this amount are considered
to belong to different clusters.

OPTIONAL ARGUMENTS

`plot:` a logical flag: if `TRUE`, a picture of declustering is drawn. The default
is `TRUE`.

VALUE
>   a declustered `"timeSeries"` or `"signalSeries"` object data. A plot
>   of the original series data, a `qplot` of the gaps, a plot of the declustered
>   series data, and a `qplot` of gaps of the declustered data will be drawn
>   on a graphical device if `plot=T`.

REFERENCES
>   Embrechts, P., Klueppelberg, C., and Mikosch, T. (1997). *Modelling
>   Extremal Events*. Springer.

SEE ALSO
>   `pot`, `exindex` .

EXAMPLE

```
# decluster the 200 exceedances of a particular threshold in
# the negative BMW data
out = pot(-bmw, ne=200)
decluster(out$data, 30)
```

dell Stock Returns and Trading Volume of Dell Corporation

SUMMARY

The `dell.s` data vector is a data set with 1261 values, representing daily stock returns of Dell Corporation in percentage points from August 24, 1993, to August 19, 1998. The `dell.v` data vector contains the daily trading volume for the same time period.

disaggregate   Time Series Disaggregation/Distribution

DESCRIPTION

Disaggregates or distributes a low frequency time series into a high frequency time series.

```
disaggregate(data, k, method="spline", how="sum", x=NULL,
             out.positions=NULL, ...)
```

REQUIRED ARGUMENTS

data : a vector, or a matrix, or a `"timeSeries"` object that represents the low frequency time series to be disaggregated.

k : a positive integer specifying the number of time periods to distribute `data` into. For example, to disaggregate an annual series into a quarterly series, you set `k` to 4.

OPTIONAL ARGUMENTS

method : a character string specifying the method for disaggregation. Valid choices are `"spline"` and `"gls"`, which stand for cubic spline and generalized least squares, respectively.

how : a character string specifying how the disaggregation will be accomplished. Valid choices are `"sum"` and `"mean"`. If `how="sum"`, each low frequency observation is equal to the sum of the corresponding high frequency observations; if `how="mean"`, each low frequency observation is equal to the average of the corresponding high frequency observations.

x : NULL, or a vector, or a two-dimensional numeric object that specifies some related variables in high frequency. The number of rows in `x` must be equal to the number of rows in `data` multiplied by `k`. If `method="gls"`, then `x` must be specified.

out.positions : a `"timeDate"` object, or a character string which can be passed to `timeDate` function, to specify the positions for the disaggregated data. It must have length `n*k`, where `n` is the number of rows in `data`. The default is `NULL`.

... : any optional arguments that can be passed to the `timeDate` function to parse the `out.positions` specification.

VALUE

a vector, or a matrix, or a `"timeSeries"` object that represents the disaggregated high frequency time series.

REFERENCES

Boot, J. C. G., Feibes, W., and Lisman, J. H. C. (1967). Further methods of derivation of quarterly figures from annual data. *Applied Statistics*, 16:65-75.

Chow, G., and Lin, A. (1971). Best linear unbiased interpolation, distribution, and extrapolation of time series by related series. *Review of Economics and Statistics*, 53:372-375.

SEE ALSO

aggregate, cspline , spline .

EXAMPLE

```
# distribute annual industrial production over quarters
ip.quarter = disaggregate(nelson.dat[,"IP"], 4)
```

DowJones30 Daily Closing Prices of Dow Jones Industrial Average
Constituents

SUMMARY
> This is a daily `"timeSeries"` object from January 2, 1991 to January
> 2, 2001, with thirty columns representing the closing prices of thirty
> stocks in Dow Jones Industrial Average.

AA: closing prices of Alcoa, Inc.

AXP: closing prices of American Express Co.

T: closing prices of AT & T Corp.

BA: closing prices of Boeing Co.

CAT: closing prices of Caterpillar, Inc.

C: closing prices of Citigroup.

KO: closing prices of Coco Cola, Co.

DD: closing prices of Du Pont Co.

EK: closing prices of Eastman Kodak.

XOM: closing prices of Exxon Mobile.

GE: closing prices of General Electric Co.

GM: closing prices of General Motors.

HWP: closing prices of Hewlett-Packard.

HD: closing prices of Home Depot Inc.

HON: closing prices of Honeywell International.

INTC: closing prices of Intel Corp.

IBM: closing prices of International Business Machine.

IP: closing prices of International Paper Co.

JPM: closing prices of JP Morgan Chase & Co.

JNJ: closing prices of Johnson & Johnson.

MCD: closing prices of McDonalds Corp.

MRK: closing prices of Merck & Co.

`MSFT:` closing prices of Microsoft Corp.

`MMM:` closing prices of 3M Company.

`MO:` closing prices of Philip Morris.

`PG:` closing prices of Procter & Gamble.

`SBC:` closing prices of SBC Communications, Inc.

`UTX:` closing prices of United Technologies Corp.

`WMT:` closing prices of Wal-mart Stores.

`DIS:` closing prices of Walt Disney Co.

d.pgram Periodogram Estimate of Fractional Integration Parameter

DESCRIPTION

Returns the estimate of the fractional integration parameter or the Hurst parameter for a fractional integrated time series.

```
d.pgram(x, spans=1, taper=0.1, pad=0, detrend=F, demean=T,
        method="ls", output="d", lower.percentage=0.1,
        minNumFreq=10, plot=F, ...)
```

REQUIRED ARGUMENTS

x : a vector, matrix, data frame, or a `"timeSeries"` object with a numeric object in the data slot, representing either a univariate or a multivariate time series.

OPTIONAL ARGUMENTS

spans : a sequence of lengths of modified Daniell smoothers to run over the raw periodogram. Use `spans=1`, the default, for the raw periodogram. A modified Daniell smoother has all values equal except for the 2 end values which are half the size of the others. The values should be odd integers.

taper : fraction of each end of the time series that is to be tapered. A split cosine taper is applied to `taper * length(x)` points at each end of series. This must take values between 0 and 0.5. The default is 0.1.

pad : fraction of the length of x that is to be padded: `pad * length(x)` zeros are added to the end of the series before computing the periodogram. The default is 0.

detrend : a logical flag: if `TRUE`, a least squares line is removed from each component of the series before computing periodogram. The default is `FALSE`.

demean : a logical flag: if `TRUE`, the mean of each series is removed before computing the periodogram (`detrend` also removes the mean). The default is `TRUE`.

method : a character string specifying the regression estimator to use. Valid choices are: `"ls"` for least squares, and `"l1"` for L1 regression. The default is `"ls"`.

output : a character string specifying the type of the output. Valid choices are: `"d"` for the fractional integration or difference parameter, and `"H"` for Hurst parameter. The default is `"d"`.

`lower.percentage` : a number specifying the percent of the lower end of peri-
odogram to be used in the regression. This must be a number between
0 and 1. The default is `0.1`.

`minNumFreq` : an integer giving the minimum number of frequencies at the
lower end of periodogram to be used in the regression. The default is
10.

`plot` : a logical flag: if `TRUE`, the log-log plot of periodogram is displayed on
a graphical device. The default is `FALSE`.

`...` : any other optional arguments that may be passed down to the `plot`
function.

VALUE

a numeric vector giving the estimates of fractional integration parame-
ter or Hurst parameter. If `plot` is `TRUE`, the log-lot plot of periodogram
is displayed on a graphical device.

DETAILS

The function first calls `spec.pgram` to obtain a periodogram estimate,
then log periodogram regression is used to estimate the fractional inte-
gration parameter, based on the asymptotic behavior of the parameter
when the frequency is close to zero.

If the input `x` is multivariate, then the fractional integration parameter
is estimated for each series separately.

REFERENCES

Beran, J. (1994). *Statistics for Long Memory Processes*, Chapman
and Hall, New York.

Taqqu, M. S., and Teverovsky, V. (1998). On estimating the intensity
of long-range dependence in finite and infinite variance time series, in
R. J. Adler, R. E. Feldman and M. S. Taqqu (eds.): *A Practical Guide
to Heavy Tails: Statistical Techniques and Applications*. Birkhauser,
Boston.

Taqqu, M. S., Teverovsky, V., Willinger, W. (1995). Estimators for
long range dependence: an empirical study. *Fractals*, 3(4):785-798.

SEE ALSO

`FARIMA`,`d.ros` ,`d.whittle` ,`gphTest` ,`rosTest` ,`spec.pgram` .

EXAMPLE

```
# Hurst parameter estimate of absolute DELL stock returns
d.pgram(abs(dell.s), output="H", plot=T)
```

d.ros R/S Estimate of Fractional Integration Parameter

DESCRIPTION

Returns the estimate of the fractional integration parameter or the Hurst parameter for a fractional integrated time series.

```
d.ros(x, minK=4, k.ratio=2, minNumPoints=3, method="ls",
     output="d", plot=F, ...)
```

REQUIRED ARGUMENTS

x : a vector, or a `"timeSeries"` object with a vector in the data slot, representing either a univariate time series.

OPTIONAL ARGUMENTS

minK : an integer specifying the minimum number of subsample size to use for calculating the R/S statistics. The default is 4.

k.ratio : a number specifying the ratio of number of subsample size in the next partition to number of subsample size in the current partition. The default is 2.

minNumPoints : an integer specifying the minimum number of R/S statistics for each valid sub-sample size. The default is 3.

method : a character string specifying the regression estimator to use. Valid choices are: `"ls"` for least squares, and `"l1"` for L1 regression. The default is `"ls"`.

output : a character string specifying the type of the output. Valid choices are: `"d"` for the fractional integration or difference parameter, and `"H"` for Hurst parameter. The default is `"d"`.

plot : a logical flag: if TRUE, the log-log plot of the R/S statistics versus sample size is displayed on a graphical device. The default is FALSE.

... : any other optional arguments that may be passed down to the plot function.

VALUE

a numeric vector giving the estimates of fractional integration parameter or Hurst parameter. If plot is TRUE, the log-lot plot of the R/S statistics versus sample size is displayed on a graphical device.

DETAILS

The function partitions the data into various numbers of blocks to compute the R/S statistics. On the log-log plot of the R/S statistics versus sample size, the slope should be equal to the Hurst parameter.

REFERENCES

Beran, J. (1994). *Statistics for Long Memory Processes*, Chapman and Hall, New York.

Taqqu, M. S., and Teverovsky, V. (1998). On estimating the intensity of long-range dependence in finite and infinite variance time series, in R. J. Adler, R. E. Feldman and M. S. Taqqu (eds.): *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*. Birkhauser, Boston.

Taqqu, M. S., Teverovsky, V., Willinger, W. (1995). Estimators for long range dependence: an empirical study. *Fractals*, 3(4):785-798.

SEE ALSO

`FARIMA,d.pgram ,d.whittle ,gphTest ,rosTest` .

EXAMPLE

```
# Hurst parameter estimate of absolute DELL stock returns
d.ros(abs(dell.s), output="H", plot=T)
```

commMat Elimination, Duplication, Commutation and Power Matrices

DESCRIPTION

Returns the commutation, duplication, elimination or power matrices.

```
commMat(m, n)
dupMat(n)
dupMatInv(n)
elimMat(n)
powMat(x, p)
```

REQUIRED ARGUMENTS

n : an integer giving the dimension of a square matrix, or the number of columns of a rectangular matrix.

m : an integer giving the number of rows of a rectangular matrix.

x : a square matrix.

p : an integer giving the power of the matrix x to be returned.

VALUE

`commMat` returns the commutation matrix, `dupMat` returns the duplication matrix, `dupMatInv` returns the inverse of duplication matrix, `elimMat` returns the elimination matrix, and `powMat` returns a power of a square matrix.

DETAILS

The commutation matrix is defined by`commMat(m,n) %*% vec(G) = vec(G')` for any matrix `G` of dimension `m x n`. The duplication matrix is defined by `dupMat(n) %*% vech(A) = vec(A)` for any symmetric matrix `A`. The inverse of the duplication matrix is defined by `dupMatInv(n) %*% vec(A) = vech(A)` for any symmetric matrix `A`. The elimination matrix is defined by `elimMat(n) %*% vec(A) = vech(A)` for any square matrix. The power matrix is defined as x^p.

REFERENCES

Lutkepohl, H. (1990). *Introduction to Multiple Time Series Analysis*, Springer-Verlag.

d.whittle Whittle Estimate of Fractional Integration Parameter

DESCRIPTION

Returns the estimate of the fractional integration parameter or the Hurst parameter for a fractional integrated time series.

```
d.whittle(x, spans=1, taper=0.1, pad=0, detrend=F,
          demean=T, output="d")
```

REQUIRED ARGUMENTS

x : a vector, matrix, data frame, or a `"timeSeries"` object with a numeric object in the data slot, representing either a univariate or a multivariate time series.

OPTIONAL ARGUMENTS

spans : a sequence of lengths of modified Daniell smoothers to run over the raw periodogram. Use `spans=1`, the default, for the raw periodogram. A modified Daniell smoother has all values equal except for the 2 end values which are half the size of the others. The values should be odd integers.

taper : fraction of each end of the time series that is to be tapered. A split cosine taper is applied to `taper * length(x)` points at each end of series. This must take values between 0 and 0.5. The default is 0.1.

pad : fraction of the length of x that is to be padded: `pad * length(x)` zeros are added to the end of the series before computing the periodogram. The default is 0.

detrend : a logical flag: if `TRUE`, a least squares line is removed from each component of the series before computing periodogram. The default is `FALSE`.

demean : a logical flag: if `TRUE`, the mean of each series is removed before computing the periodogram (`detrend` also removes the mean). The default is `TRUE`.

output : a character string specifying the type of the output. Valid choices are: `"d"` for the fractional integration or difference parameter, and `"H"` for Hurst parameter. The default is `"d"`.

VALUE

a numeric vector giving the estimates of fractional integration parameter or Hurst parameter.

DETAILS

The function first calls `spec.pgram` to obtain a periodogram estimate, then uses Whittle's method to estimate the fractional integration parameter of a FARIMA(0,d,0) process.

If the input `x` is multivariate, then the fractional integration parameter is estimated for each series separately.

REFERENCES

Beran, J. (1994). *Statistics for Long Memory Processes*, Chapman and Hall, New York.

Taqqu, M. S., and Teverovsky, V. (1998). On estimating the intensity of long-range dependence in finite and infinite variance time series, in R. J. Adler, R. E. Feldman and M. S. Taqqu (eds.): *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*. Birkhauser, Boston.

Taqqu, M. S., Teverovsky, V., Willinger, W. (1995). Estimators for long range dependence: an empirical study. *Fractals*, 3(4):785-798.

SEE ALSO

FARIMA,`d.ros` ,`d.pgram` ,`gphTest` ,`rosTest` ,`spec.pgram` .

EXAMPLE

```
# Hurst parameter estimate of absolute DELL stock returns
d.whittle(abs(dell.s), output="H")
```

empirical.copula.object Empirical Copula Class Object

DESCRIPTION
>   These are S version 4 objects of class `"empirical.copula"`.

GENERATION
>   This class of objects is constructed from the `empirical.copula(x, y)` function.

METHODS
>   The `"empirical.copula"` class of objects currently has methods for the following generic functions:
>
>   `Afunc`, `LAMBDA`, `pcopula`, `contour.plot`, `tail.index`, `Kendalls.tau`, `Spearmans.rho`.
>
>   Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula`, `contour.dcopula`, `contour.pcopula`.

STRUCTURE
>   The following slots must be present in a legitimate `"empirical.copula"` class object:

>   x : data points that are assumed to have a uniform(0,1) marginal distribution. It must be either a vector, a list or a matrix. If `x` is a vector, then a vector `y` of the same length should be supplied. The data is then assumed to be `(x,y)`. If it is a list, the first two components of the list should represent the data points and have the same length. It is is a matrix, its first two columns are assumed to contain the data points.

>   y : same type of object as `x`.

DETAILS
>   If some of the input data points are negative or greater than one, a warning is produced. All such points are disregarded.

REFERENCES
>   Nelsen, R. B. (1999). *An Introduction to Copulas*. New York: Springer-Verlag.

SEE ALSO
>   `copula.object`.

emplot Plot of Empirical Distribution Function

DESCRIPTION

        Generates a plot of the empirical distribution function of a sample.

```
emplot(data, alog="x", labels=T)
```

REQUIRED ARGUMENTS

  `data` : a numeric vector or a `"timeSeries"` object with a numeric vector in the data slot.

OPTIONAL ARGUMENTS

  `alog` : controls logarithmic axes. Values `"xy"`, `"x"`, or `"y"` produce log-log or log-x or log-y axes. (This causes the corresponding usr coordinates to be in units of log base 10.)

  `labels` : a logical flag: if `TRUE`, both axes are labeled. The default is `TRUE`.

DETAILS

        This is a simple explanatory function. A straight line on the double log scale indicates Pareto tail behavior. A plot of the empirical distribution will be drawn on a graphical device.

SEE ALSO

        `qplot,meplot` .

EXAMPLE

```
# Danish fire insurance data show Pareto tail behavior
emplot(danish)
```

ev.copula.object   Extreme Value Copula Class Object

DESCRIPTION
>   These are S version 4 objects of class `"ev.copula"` , which inherits
>   from `copula` class.

GENERATION
>   Sub-class objects of class Extreme Value copula class can be con-
>   structed from the `ev.copula(family, param)` function.

STRUCTURE
>   The following slots must be present in a legitimate `"ev.copula"` class
>   object:

`parameters` : a vector with values of the parameters for this particular instance
>   of the object `copula` (the length of this vector is equal to the number
>   of parameters allowed for this particular family).

`param.names` : typical greek letters that are used for the parameters (such as
>   `delta` or `theta`).

`param.lowbnd` : a vector with the same length as the `parameters` vector, con-
>   taining the values of the lower bounds for the parameter(s).

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for
>   the parameter(s).

`message`: a message specifying the name of the parametric copula family, and
>   the copula sub-class, such as Extreme Value copula, if applicable.

REFERENCES
>   Joe, H. (1999). *Multivariate Models and Dependence Concepts*. Lon-
>   don: Chapman & Hall.

SEE ALSO
>   `copula.object`,`ev.copula` .

ev.copula   Construction of a specific Extreme Value Copula Class Object

DESCRIPTION
>   Constructs a certain Extreme Copula class object with its correspond-
>   ing parameter vector.

>   `ev.copula(family="BB5", param=c(0.8, 1.43))`

REQUIRED ARGUMENTS

`family` : a character string defining which Extreme Value copula class to gen-
>   erate.

`param` : a numeric vector specifying the value for the parameters of this par-
>   ticular Extreme Value copula class.

VALUE
>   an object of certain Extreme Value Copula class specified by `family`
>   argument (which inherits from `copula` and `ev.copula` classes).

DETAILS
>   Each Extreme Value copula sub-class can also be constructed directly.

SEE ALSO
>   `ev.copula.object`,`archm.copula` .

EWMA Exponentially Weighted Moving Average

DESCRIPTION

Returns the exponentially weighted moving average of a given time series.

```
EWMA(x, n=9, lambda=(n - 1)/(n + 1), start="average", na.rm=F)
```

REQUIRED ARGUMENTS

x : a vector, matrix, or a "timeSeries" object.

OPTIONAL ARGUMENTS

n : a positive integer that determines the approximuate number of periods or terms to use in the moving average. This integer is used to calculate lambda, the exponential weight. If lambda is specified, then n is ignored. The default uses a 9-term EWMA.

lambda : a positive number between 0 and 1, which determines the exponential weight. If lambda is specified, then n is ignored.

start : a character string or a number that specifies the first number to use to start the EWMA filter. If start is a character string, the valid choices are: "average", and "first". When start="average", the average of the first n observations is used as the starting value; when start="first", the first observation in x is used as the starting value. If start is a number, then it will be used as the starting value. The default is "average".

na.rm : a logical flag: if TRUE, missing values will be removed before computing EWMA. The default is FALSE.

VALUE

a vector, matrix, or "timeSeries" object with the same dimension as x, which is the n-term EWMA of x.

DETAILS

The EWMA of a time series y at time t is calculuated as the weighted sum of the EWMA at time t-1 and the value of y at timet .

If there are more than one column in x, then EWMAs are computed column-wise. In this case, a vector of lambda can be specified, with each value corresponding to each column. However, the starting values with the same length must also be given.

SEE ALSO

SMA,iEMA ,iMA .

EXAMPLE

```
bond = nelson.dat[positions(nelson.dat) > timeDate("01/01/1899"), "BND"]
plot(bond, EWNA(bond, lambda=0.8))
```

exindex Estimate and Plot Extremal Index

DESCRIPTION
>   Estimates the extremal index using the blocks methods and generates
>   a plot optionally.

>   ```
>   exindex(data, block, start=5, end=NA, plot=T, reverse=F,
>           auto.scale=T, labels=T, ...)
>   ```

REQUIRED ARGUMENTS

data : a numeric vector or `"timeSeries"` object (raw values not block maxima).

block : an integer specifying the block size (the number of data values in each successive block) or a character string that takes the value of `"month"`, `"quarter"`, `"semester"` or `"year"` (if the data is a `"timeSeries"` object with calendar times).

OPTIONAL ARGUMENTS

start : an integer specifying the lowest value of K to start with, the number of blocks in which a specified threshold is exceeded. The default is 5.

end : an integer specifying the highest value of K to end with, or `NA`, in which case all blocks are used. The default is `NA`.

plot : a logical flag: if `TRUE`, a plot is generated. The default is `TRUE`.

reverse : a logical flag: if `TRUE` the plot is in terms of t increasing thresholds; if `FALSE` the plot is in terms of increasing number of blocks in which a threshold is exceeded. The default is `FALSE`.

auto.scale : a logical flag: if `TRUE`, the plot is automatically scaled; if `FALSE`, `xlim` and `ylim` graphical parameters may be entered. The default is `TRUE`.

labels : a logical flag: if `TRUE`, axes are labeled. The default is `TRUE`.

... : any optional arguments that can be passed down to the `plot` function.

VALUE
>   a matrix which contains the following columns:

N : the number of data points in which a threshold is exceeded.

K : the number of blocks in which a threshold is exceeded.

threshold : the threshold.

theta : the ratio between K and N.

exindex : the extremal index. A plot of extreme indices over increasing thresholds or numbers of blocks will be drawn on a graphical device if `plot=T`.

REFERENCES

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997). *Modelling Extremal Events*, Chapter 8, 413-429. Springer.

SEE ALSO

gev, hill .

EXAMPLE

```
# calculate extremal index for the right and left tails of the BMW
# log returns
exindex(bmw, 100)
exindex(-bmw, 100)
```

factors.mfactor   Use `factors()` on an `mfactor` Object

DESCRIPTION
> This is a method for the function `factors()` for objects inheriting
> from class `"mfactor"`. See `factors` for the general behavior of this
> function and for the interpretation of `object`.

>   factors.mfactor(object, index)

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"mfactor"`.

OPTIONAL ARGUMENTS

`index` : an integer vector, or a character vector giving the indices of the factors
> to be returned. If `index` is a character vector, it must correspond to
> the names of the factors.

VALUE
> a vector or a matrix representing the factor returns or factor scores
> from the fitted multi-factor model.

SEE ALSO
> `factors`,`mfactor.r2` .

factors   Factor Returns/Scores from Multi-Factor Models

DESCRIPTION

Returns the selected factor returns or factor scores from a fitted multi-factor model.

This function is an S Version 3 generic (see `Methods`); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: `mfactor`.

```
factors(object, ...)
```

REQUIRED ARGUMENTS

`object` : any object representing a fitted model. It usually contains a component called "`factors`.

VALUE

a vector or a matrix representing the factor returns or factor scores.

SEE ALSO

`loadings`,`loadings.mfactor` ,`mfactor` .

FARIMA.d2ar Autoregressive Representation of a Fractional Integrated Noise

DESCRIPTION

Returns the autoregressive representation of a fractional integrated noise.

```
FARIMA.d2ar(nAR, d, kapprox=100)
```

REQUIRED ARGUMENTS

nAR : the required order of the autoregressive representation plus 1. So if you want to return the AR(1) representation, you set nAR to 2.

d : a number between -0.5 and 0.5 which specifies the fractional difference parameter.

OPTIONAL ARGUMENTS

kapprox : a large integer specifying the maximal order for which the exact formula is used. For the AR coefficients with orders greater than kapprox, the asymptotic approximation is used.

VALUE

a numeric vector with length nAR giving the autoregressive representation, with the first element being 1.

REFERENCES

Beran, J. (1994). *Statistics for Long Memory Processes*, Chapman and Hall, New York.

Hosking, J. R. M. (1981). Fractional differencing. *Biometrika*, 68(1):165-176.

SEE ALSO

arma2ma,VAR.ar2ma .

EXAMPLE

```
# returns AR(9) representation of d=0.3
FARIMA.d2ar(10, 0.3)
```

FARIMA.fit Fitting of Fractional ARIMA Models

DESCRIPTION

Called by `FAR` and `FARIMA` functions to fit the models. It is not supposed to be called by the users directly.

`FARIMA.fit(xInput, mmax, p, q, M=100)`

REQUIRED ARGUMENTS

`xInput` : a numeric vector giving the time series to be modeled.

`mmax` : a positive integer that specifies the maximal number of difference to apply to the time series `xInput` before fitting a stationary fractional AR model. Currently this must be either 0 or 1. If you only want to consider stationary models, set `mmax` to 0.

`p` : a positive integer specifying the order of the autoregressive part.

`q` : a positive integer specifying the order of the moving average part.

OPTIONAL ARGUMENTS

`M` : a large integer specifying the number of terms in the truncated series used to approximate the likelihood function. The default is 100.

FARIMA.object Fractional ARIMA Model Objects

DESCRIPTION

These are objects of class `"FARIMA"` and represent the fit of a fractional ARIMA model.

GENERATION

This class of objects is returned from the `FAR` function or `FARIMA` function.

METHODS

The `"FARIMA"` class of objects currently has methods for the following generic functions:

`coef`, `residuals`, `fitted`, `vcov`, `plot`, `predict`, `print`, `summary`.

STRUCTURE

The following components must be present in a legitimate `"FARIMA"` object:

call : an image of the call that produced the object.

model : a list with the following named components: `"d"` which is the estimated fractional difference parameter, `"ar"` which is the estimated AR coefficients, `"ma"` which is the estimated MA coefficients, `"mean"` which is the unconditional mean of the fitted stationary model, and `"sigma2"` which is the residual variance estimate.

m : an integer which is either 0 or 1. If m=0, the FARIMA model is fitted to the original time series; if m=1, the FARIMA model is fitted to the first difference of the original time series. In the latter case, the `"mean"` component of `model` is the coefficient of the linear trend term.

delta : a number between -0.5 and 0.5, which is the estimate of the fractional difference parameter after taking consideration of m. In general, the `"d"` component of `model` is equal to m+delta.

n.used : the number of sample size used to fit the model after adjusting for the conditioning observations.

BIC : the Bayesian Information Criterion of the returned model.

loglike : the log-likelihood value of the returned model.

residuals : the residuals from the returned model.

fitted : the fitted values from the returned model. If m=0, this is the fitted values for the original time series; if m=1, this is the fitted values for the first difference of the original time series.

`BIC.all` : a matrix giving the BIC values for all the models considered. This is returned only if both `p` and `q` are `NULL` in the original function call.

`cov` : the covariance matrix of the estimated model parameters.

`CI` : a matrix with two columns giving the confidence intervals of the estimated model parameters. This is returned only if `CI=T` in the original function call.

SEE ALSO

`FAR, FARIMA` .

FARIMA.spec Theoretical Spectral Density of FARIMA Models

DESCRIPTION

Returns the theoretical spectral density of a FARIMA model.

```
FARIMA.spec(model, circfreq)
```

REQUIRED ARGUMENTS

`model` : a list with the following named components: `"d"` which is the esti-
mated fractional difference parameter, `"ar"` which is the estimated
AR coefficients, `"ma"` which is the estimated MA coefficients, and
`"sigma2"` which is the residual variance estimate.

`circfreq` : a numeric vector giving the frequencies (cycles/sample) for which
the spectral densities are to be computed.

VALUE

a list with the following components:

`freq` : a numeric vector giving the frequencies (cycles/sample) for which the
spectral densities are computed.

`f` : a numeric vector giving the spectral densities corresponding to `freq`.

`f.long` : a numeric vector giving the long memory part of `f`.

`f.short` : a numeric vector giving the short memory part of `f`.

REFERENCES

Beran, J. (1994). *Statistics for Long Memory Processes*, Chapman
and Hall, New York.

FARIMA Fit a Fractional ARIMA Model

DESCRIPTION

Fits a fractional ARIMA model to a univariate time series.

```
FARIMA(x, p=NULL, q=NULL, p.range=c(0, 2), q.range=c(0, 2), mmax=1,
        alpha=0.05, CI=T, na.rm=F, M=100, kapprox=100, trace=T, ...)
```

REQUIRED ARGUMENTS

x : a numeric vector, or a `"timeSeries"` object with a numeric data slot.

OPTIONAL ARGUMENTS

p : a positive integer specifying the order of the autoregressive part. If `p` and `q` are `NULL`, then the order of the ARIMA model is determined by searching for `p` and `q` in the grid specified by `p.range` and `q.range`, using Bayesian Information Criterion (BIC).

q : a positive integer specifying the order of the moving average part. If `p` and `q` are `NULL`, then the order of the ARIMA model is determined by searching for `p` and `q` in the grid specified by `p.range` and `q.range`, using Bayesian Information Criterion (BIC).

p.range : a vector with two positive integers that specifies the range for the order of the autoregressive part. This is ignored if `p` and `q` are given.

q.range : a vector with two positive integers that specifies the range for the order of the moving average part. This is ignored if `p` and `q` are given.

mmax : a positive integer that specifies the maximal number of difference to apply to the time series `x` before fitting a stationary fractional AR model. Currently this must be either 0 or 1. If you only want to consider stationary models, set `mmax` to 0. The default is set to be 1.

alpha : a number specifying the level of significance. In particular, `1-alpha` is the confidence level of the confidence intervals for the estimated parameters. This is only used if `CI=T`. The default is 0.05.

CI : a logical flag: if `TRUE`, confidence intervals for the estimated parameters are saved in the returned object. The default is `TRUE`.

na.rm : a logical flag: if `TRUE`, missing values will be removed before fitting the model. The default is `FALSE`.

M : a large integer specifying the number of terms in the truncated series used to approximate the likelihood function. The default is 100.

kapprox : a large integer specifying the maximal order for which the exact formula is used for computing the AR representation of a fractional integrated noise. For the AR coefficients with orders greater than `kapprox`, the asymptotic approximation is used. This is only used in computing the model residuals. The default is 100.

trace : a logical flag: if `TRUE`, a message will be printed on the screen when the function searches for the best order over `p.range` and `q.range`. This is only used if `p` and `q` are `NULL`. The default is `TRUE`.

VALUE

an object of class `"FARIMA"` representing the fit of the fractional ARIMA model. See `FARIMA.object` for details.

REFERENCES

Beran, J. (1994). *Statistics for Long Memory Processes*, Chapman and Hall, New York.

Beran, J. (1995). Maximum likelihood estimation of the differencing parameter for invertible short and long memory ARIMA models. *Journal of Royal Statistical Society B*, 57(4):659-672.

Haslett, J., and Raftery, A. E. (1989). Space-time modeling with long-memory dependence: assessing Ireland's wind power resource (with discussion). *Applied Statistics*, 38:1-50.

SEE ALSO

`arima.fracdiff`, `FAR` ,`FARIMA.object` ,`SEMIFAR` .

EXAMPLE

```
set.seed(2)
ts.sim <- simulate.FARIMA(list(d=.3, ar=.5, ma=0.1, mean=1), n=3000)
FARIMA(ts.sim, p=1, q=1, mmax=0)
```

FAR Fit a Fractional Autoregressive Model

DESCRIPTION

Fits a fractional autoregressive model to a univariate time series.

```
FAR(x, p=NULL, p.range=c(0, 2), mmax=1, alpha=0.05, CI=T, M=100,
    kapprox=100, na.rm=F, trace=T, ...)
```

REQUIRED ARGUMENTS

x : a numeric vector, or a `"timeSeries"` object with a numeric data slot.

OPTIONAL ARGUMENTS

p : a positive integer specifying the order of the autoregressive model. If p is NULL, then the order is determined by searching for p in the range specified by p.range, using Bayesian Information Criterion (BIC).

p.range : a vector with two positive integers that specifies the range for the order of the autoregressive model. This is ignored if p is given.

mmax : a positive integer that specifies the maximal number of difference to apply to the time series x before fitting a stationary fractional AR model. Currently this must be either 0 or 1. If you only want to consider stationary models, set mmax to 0. By setting mmax to 1, FAR will try both values and select the one which yields the smaller BIC value. The default is set to be 1.

alpha : a number specifying the level of significance. In particular, 1-alpha is the confidence level of the confidence intervals for the estimated parameters. This is only used if CI=T. The default is 0.05.

CI : a logical flag: if TRUE, confidence intervals for the estimated parameters are saved in the returned object. The default is TRUE.

M : a large integer specifying the number of terms in the truncated series used to approximate the likelihood function. The default is 100.

kapprox : a large integer specifying the maximal order for which the exact formula is used for computing the AR representation of a fractional integrated noise. For the AR coefficients with orders greater than kapprox, the asymptotic approximation is used. This is only used in computing the model residuals. The default is 100.

na.rm : a logical flag: if TRUE, missing values will be removed before fitting the model. The default is FALSE.

`trace` : a logical flag: if `TRUE`, a message will be printed on the screen when the function searches for the best order over `p.range`. This is only used if `p` is `NULL`. The default is `TRUE`.

VALUE

an object of class `"FARIMA"` representing the fit of the fractional AR model. See `FARIMA.object` for details.

REFERENCES

Beran, J. (1994). *Statistics for Long Memory Processes*, Chapman and Hall, New York.

Beran, J. (1995). Maximum likelihood estimation of the differencing parameter for invertible short and long memory ARIMA models. *Journal of Royal Statistical Society B*, 57(4):659-672.

Haslett, J., and Raftery, A. E. (1989). Space-time modeling with long-memory dependence: assessing Ireland's wind power resource (with discussion). *Applied Statistics*, 38:1-50.

SEE ALSO

`arima.fracdiff`, `FARIMA` ,`FARIMA.object` ,`SEMIFAR` .

EXAMPLE

```
set.seed(2)
ts.sim <- simulate.FARIMA(list(d=.3, ar=.2, mean=1), n=3000)
FAR(ts.sim, p=1, mmax=0)
```

fbar.exp.portf Calculation of Portfolio Value at Risk

DESCRIPTION

> The function `fbar.exp.portf` calculates the probability that the re-
> turn of a two-asset portfolio exceeds a certain lower threshold. The
> functions `VaR.exp.portf` and `VaR.exp.sim` calculate the Value-at-
> Risk (VaR) and Expected Shortfall of a two-asset portfolio for a given
> probability using analytical methods and simulations method, respec-
> tively.


> ```
> fbar.exp.portf(a, copula, x.est, y.est, lambda1, lambda2,
>     q=NA, subdivisions=1000, tol.f=5e-5)
> VaR.exp.portf(Q, copula, x.est, y.est, lambda1, lambda2,
>     range, subdivisions=1000, tol.f=5e-5)
> VaR.exp.sim(n, Q, copula, x.est, y.est, lambda1, lambda2 )
> ```

REQUIRED ARGUMENTS

> `a`: a numeric value specifying the portfolio return (the quantile).

> `Q`: the probability that Value-at-Risk is larger than the portfolio return
> (or the quantile).

> `n`: the numeric value specifying the number of simulations performed to
> calculate Value-at-Risk.

`copula`: an object of class `"copula"`.

`x.est`: an object of class `"gpd"` fitted to the first asset in the portfolio.

`x.est`: an object of class `"gpd"` fitted to the second asset in the portfolio.

`lambda1`: a numerical value specifying the weight of the first asset in the port-
folio.

`lambda2`: a numerical value specifying the weight of the second asset in the
portfolio.

OPTIONAL ARGUMENTS

> `q`: probability assessment (if any) that the portfolio return will exceed
> the lower threshold. Default is `NA`.

`subdivisions`: a numerical value specifying the subdivisions argument in S-
PLUS function `integrate`. The default is 1000.

`tol.f`: a numerical value specifying the `abs.tol` argument in S-PLUS func-
tion `integrate`. The default is `5e-5`.

VALUE

the probability that return of a two-asset portfolio exceeds a certain lower threshold or the Value-at-Risk (VaR(Q)) of a two-asset portfolio for a given probability. If using `VaR.exp.sim`, Expected Shortfall is calculated in addition to VaR.

DETAILS

This set of function assumes the portfolio return (r) equals:`r = log(lambda1*exp(x) + lambda2*exp(y))` , where `x` and `y` are asset 1 and asset 2 return, respectively.

SEE ALSO

`gpd.tail.`

EXAMPLE

```
tmp = gpd.tail(UTI.ELEC[,2], upper=0.006, lower=-0.006, plot=F)
tmp1 = gpd.tail(UTI.GAS[,2], upper=0.006, lower=-.006, plot=F)
tmp2 = fit.copula(empirical.copula(gpd.2p(UTI.ELEC[,2],tmp),
                  gpd.2p(UTI.GAS[,2],tmp1)), "bb1", plot=F)
fbar.exp.portf(0.01714058575216979, copula=tmp2, x.est=tmp, y.est=tmp1,
               lambda1=0.5, lambda2=0.5)

VaR.exp.portf(0.01, range=c(0.016,0.08), copula=tmp2, x.est=tmp,
              y.est=tmp1, lambda1=0.5, lambda2=0.5)

VaR.exp.sim(n=10000, Q=0.01, copula=tmp2, x.est=tmp, y.est=tmp1,
            lambda1=0.5, lambda2=0.5)[2]
```

fevDec Forecast Error Variance Decomposition

DESCRIPTION
>   Returns the forecast error variance decomposition given a fitted VAR
>   object.

>   `fevDec(x, period=NULL, std.err="none", plot=F, unbiased=F,`
>   `        order=NULL, ...)`

REQUIRED ARGUMENTS

>   x : an object inheriting from class `"VAR"`.

OPTIONAL ARGUMENTS

  period : the number of periods to compute the forecast error variance de-
  composition. By default, we set `period=p*(k-1) + 1`, where p is the
  order of the autoregressive model, and k is the dimension of the mul-
  tivariate response.

  std.err : a character string specifying the type of standard errors to be re-
  turned. Currently the only valid choices are `"none"` and `"asymptotic"`:
  if `std.err="none"`, standard errors will not be computed; if `std.err="asymptotic"`,
  the asymptotic standard errors will be returned. The default is `"none"`.

  plot : a logical flag: if `TRUE`, the forecast error variance decomposition will
  be plotted. The default is `FALSE`.

  unbiased : a logical flag; if `TRUE`, unbiased estimate of error covariance will be
  used; if `FALSE`, the maximum likelihood estimate of error covariance
  will be used. The default is `TRUE`.

  order : a numeric index vector, or a character vector with elements corre-
  sponding to the variable names used to fit x. This can be used to
  rearrange the order of the variables when computing forecast error
  variance decomposition.

  ... : any optional arguments that may be passed to `plot.impDecomp`.

VALUE
>   an S Version 3 object of class `"impDecomp"`, containing the following
>   components:

  values : an array with dimension `c(k, k, period)` representing the forecast
  error variance decomposition, where k is the dimension of the multi-
  variate response.

`std.err` : an array with dimension `c(k, k, period)` representing the standard errors of forecast error variance decomposition, where `k` is the dimension of the multivariate response. This is only present if `std.err="asymptotic"` in the call. A plot of the forecast error variance decomposition will be shown in a graphical device if `plot=T`.

REFERENCES

Lutkepohl, H. (1990). *Introduction to Multiple Time Series Analysis*, Springer-Verlag.

SEE ALSO

`impRes`, `plot.impDecomp` , `VAR.ar2ma` .

EXAMPLE

```
pol.mod = VAR(cbind(M2,GDP,U)~ar(13), data=policy.dat)
pol.fev = fevDec(pol.mod, period=5, plot=T)
```

fgarch.control Control Parameters for FGARCH Estimation

DESCRIPTION

   Allows the users to set values affecting the algorithm used for FGARCH
   estimation in `fgarch` function.

```
fgarch.control(iter.bfgs=100, iter.ls=100, lag=1000,
               tolg=1e-05, tolx=1e-05, toll=0.1,
               epsilon=1.49011611938477e-08)
```

OPTIONAL ARGUMENTS

`iter.bfgs:` the maximum number of iterations allowed in the BFGS algorithm.
   The default is 100.

`iter.ls:` the maximum number of iterations allowed in the line search. The
   default is 100.

`lag:` the lag truncation used to approximate fractional integration. The
   default is 1000.

`tolg:` convergence tolerance in gradient zeroing. The default is `1e-5`.

`tolx:` convergence tolerance in function evaluation. The default is `1e-5`.

`toll:` precision in line minimization. The default is `0.1`.

`epsilon:` machine precision.

VALUE

   a list containing the values used for each of the control parameters.

SEE ALSO

   `bhhh.control`.

EXAMPLE

```
hp.fgarch = fgarch(hp.s~1, ~fiegarch(1,1),
    control=fgarch.control(lag=1500))
```

fgarch.eq Function to Interpret Conditional Variance Specification

DESCRIPTION

Translates `fgarch` variance equation.

        `fgarch.eq(x)`

REQUIRED ARGUMENTS

x: one of the following two specifications of a conditional variance struc-
ture in a fractionally integrated GARCH model: '~figarch(p,q)', '~fiegarch(p,q)'.
Exogenous variables are also allowed.

VALUE

a model list specifying the particular conditional variance structure.

SEE ALSO

`garch.eq`, `mgarch.eq`.

fgarch.model Create FGARCH Model List

DESCRIPTION

Converts formula specifications into a model list that can be used by the functions `fgarch`. Used primarily by `fgarch` and by users to create a model structure to edit to obtain particular model parameter values.

```
fgarch.model(formula.mean, formula.var)
```

REQUIRED ARGUMENTS

`formula.mean`: a formula specifying the conditional mean equation structure of ARMA form. Can have the form y~arma(p,q), where `y` is the response variable. The response variable `y` can be present or absent. If y does not appear in `formula.mean`, it must be specified by `series=y`. `arma(p,q)` can be replaced by `ar(p)`, `ma(q)`. A constant mean term is included by y~1 or 1+arma(p,q), etc. Exogenous variables are also allowed in the formula.

`formula.var`: a formula specifying a fractionally integrated GARCH model to fit, namely either '~figarch(p,q)' or '~fiegarch(p,q)'.

VALUE

an object of class `"fgarch.model"`.

The conditional mean equation is specified by

`c.which` : a logical variable. `TRUE` indicates a constant is included in the conditional mean equation.

`c.value` : a scalar, the initial value for the constant term in the conditional mean equation.

AR: a list containing the following components: `order`, giving the order of the AR component; and `value`, a numerical vector giving the initial values of the AR coefficients.

MA: a list containing the following components: `order`, giving the order of the MA component; and `value`, a numerical vector giving the initial values of the MA coefficients.

X: a list containing the following components: `term`, giving the term labels for exogenous variables in the mean equation; and `value` , a numerical vector giving the initial values of the coefficients for exogenous variables.

The conditional variance equation is specified by

a.value: a scalar giving the initial value for the constant term in the variance equation.

arch: a list containing the following components: `order`, giving the order of the garch moving average; `value`, a numeric vector giving the initial values of the coefficients; and `leverage$value`, a numeric vector of length `order` giving the initial values of the leverage effects.

garch: a list containing the following components: `order`, giving the order of the GARCH autoregression; and `value`, a numeric vector of length `order` giving the initial values of the coefficients.

Z: a list containing the following components: `term`, giving the term labels for exogenous variables in the variance equation; and `value`, a numerical vector giving the initial values of the coefficients for exogenous variables.

d : the initial value of the order of fractional integration.

Internal model names not needed by the users.

imod : an integer with value 0 or 1, specifying fractionally integrated FI-GARCH and FIEGARCH respectively.

inpar : integer, 1 indicating to use the internal program to set default initial values. Otherwise, use the user supplied initial values.

SEE ALSO

`garch.model`, `mgarch.model`.

fgarch Fit Univariate Fractionally Integrated GARCH Models

DESCRIPTION

Fits univariate fractionally integrated GARCH models in either FI-GARCH or FIEGARCH formulation as in Bollerslev and Mikkelsen (1996).

```
fgarch(formula.mean=~ arma(0,0), formula.var=~ figarch(1,1),
       series=NULL, series.start=1, x.start=1, xlag=0,
       z.start=1, leverage=F, model=NULL, trace=T,
       control=NULL)
```

REQUIRED ARGUMENTS

formula.mean: a formula specifying the conditional mean equation structure of ARMA form. Can have the form y~arma(p,q), where y is the response variable. The response variable y can be present or absent. If y does not appear in `formula.mean`, it must be specified by the argument `series`. `arma(p,q)` can be replaced by `ar(p)`, `ma(q)`. A constant mean term is removed by y~-1 or -1+arma(p,q), etc. Exogenous variables are also allowed.

formula.var: a formula specifying a fractionally integrated GARCH model to fit, namely either '~figarch(p,q)' or '~fiegarch(p,q)'. Exogenous variables are also allowed.

series: name of the response variable.

model: a list specifying the model as returned by `fgarch.model`. The required arguments are `formula.mean` and `formula.var`, or `model`. If both `formula.mean` and `formula.var` are specified by the user, `fgarch` will call `fgarch.model` to generate a model list specification. If one but not both of `formula.mean` or `formula.var` is missing, the missing equation will be specified as the default: `arma(0,0)` for the conditional mean equation, and `figarch(1,1)` for the conditional variance equation. If neither `formula.mean` nor `formula.var` is specified by the user, then a model list with the components specified the same way as in the model object created by `fgarch.model` must be supplied through the `model` argument, which has the default NULL. If the response variable y in the `formula.mean` is not specified, then it must be supplied in the argument `series`, which also has the default value NULL.

OPTIONAL ARGUMENTS

series.start: the element of `series` to begin with. For example, if `series.start=10`, the observations of `series` will start from the tenth element of `series`.

x.start: the element to begin with for exogenous variables in the mean. For example, if `x.start=10`, the observations will start from the tenth row of exogenous variables.

xlag: lag steps for exogenous variables in the mean equation.

z.start: the element to begin with for exogenous variables in the variance. For example, if `z.start=10`, the observations will start from the tenth row of exogenous variables.

leverage: a logical flag: if `TRUE`, leverage terms will be included for FIEGARCH models. The default is `FALSE`.

trace: a logical flag: if `TRUE`, step-by-step optimization results will be printed on the screen, otherwise see `opt.index` for convergence information. The default is `TRUE`.

control: a list of control parameters to be used in the numerical algorithms. See `fgarch.control` for the possible control parameters and their default settings.

VALUE

an object of class `"fgarch"`, including the following components:

series : a numeric vector, or a `"timeSeries` object, which represents the observed time series.

coef: the estimated coefficients in the model.

likelihood : the log-likelihood value evaluated at the optimum.

residuals : a numeric vector, or a `"timeSeries"` object, which represents the residuals from the fitted model.

sigma.t : a numeric vector, or a `"timeSeries"` object, which represents the estimated conditional standard deviation sequence.

model : the model list created and returned by the function `fgarch.model` , or a model list supplied by the user as an input to `fgarch` function.

opt.index : convergence indicator:

0 indicating convergence in function value has been reached.

1 indicating convergence in gradient has been reached.

2 indicating maximum number of iterations for optimization has been exceeded.

3 indicating maximum number of iterations for line search has been exceeded.

REFERENCES

Baillie, R. T., Bollerslev, T., and Mikkelsen, H. O. (1996). Fractionally integrated generalized autoregressive conditional heteroscedasticity. *Journal of Econometrics*, 74:3-30.

Bollerslev, T., and Mikkelsen, H. O. (1996). Modeling and pricing long memory in stock market volatility. *Journal of Econometrics*, 73:151-184.

SEE ALSO

`fgarch.control`, `garch` , `mgarch` .

EXAMPLE

```
hp.fgarch = fgarch(hp.s~1, ~fiegarch(1,1))
```

fgarch.unlis Unlist a `fgarch.model` Object

DESCRIPTION

Converts a list of class `"fgarch.model"` into a vector. It is not supposed to be called by the users directly.

```
fgarch.unlis(x)
```

fiegarch.init Initialization of Fractionally Integrated GARCH Models

DESCRIPTION

Initializes fractionally integrated GARCH models. It is called by `fgarch` and not supposed to be called by the users directly.

```
fiegarch.init(series, X=NULL, Cm=1, P=0, Q=0, p=1, q=1, leverage=F,
    Z=NULL, series.start=1, x.start=1, xlag=0)
figarch.init(series, X=NULL, Cm=1, P=0, Q=0, p=1, q=1, Z=NULL,
    series.start=1, x.start=1, xlag=0)
```

findthresh Find Threshold

DESCRIPTION

Finds a threshold in the data set so that a given number of extremes lie above the threshold.

```
findthresh(data, ne)
```

REQUIRED ARGUMENTS

data : a numeric vector, or a `"timeSeries"` object with a numeric vector in the data slot.

ne : an integer vector giving number of extremes above the threshold.

VALUE

a numeric vector giving the suitable thresholds.

DETAILS

When the data are tied a threshold is found so that at least the specified number of extremes lie above.

SEE ALSO

`meplot`,`hill` , `gpd` , `pot` .

EXAMPLE

```
# Find threshold giving (at least) fifty exceedances for
# danish data set
findthresh(danish, 50)
```

fit.copula Maximum Likelihood Parameter Estimation for Copulas

DESCRIPTION
> Computes ML parameter estimates for a parametric copula family.

>  ```
>  fit.copula(data, family="normal", plot=T, init.est=NA,
>      epsilon=1e-5, ...)
>  ```

REQUIRED ARGUMENTS

 `data` : an object of class `"empirical.copula"` containing the data points.

OPTIONAL ARGUMENTS

 `family` : a character string with the name of a parametric family of copulas
> to fit into the data. The following values are implemented: `"normal"`,
> `"frank"`, `"kimeldorf.sampson"`, `"gumbel"`, `"galambos"`, `"husler.reiss"`,
> `"tawn`, `"bb1"`, `"bb2"`, `"bb3"`, `"bb4"`, `"bb5"`, `"bb6"`, `"bb7"`, `"normal.mix"`,
> `"joe"`.

 `plot` : a logical value specifying if a contour plot of `empirical.copula` is
> generated. The default is `TRUE`.

 `init.est` : an initialization point for optimization routine in the MLE. If no
> argument is specified, a point with roughly same Kendall's tau is used
> to initialize the MLE. The default is `NA`.

 `epsilon` : the relative tolerance of the optimization routine in ML. The default
> value is `10e-5`.

 VALUE
> an object of class `family.copula`, with the value of the `parameters`
> slot set to the ML parameters estimate.

DETAILS
> Splus standard convex optimization function `nlminb` is used to find the
> maximum of the log-likelihood function. The final message from this
> method (regarding convergence or not) is printed into the standard
> output.

 REFERENCES
> Joe, H. (1997). *Multivariate Models and Dependence Concepts*. Lon-
> don: Chapman & Hall.

SEE ALSO
> `copula.object`, `gpdbiv` .

EXAMPLE
>  ```
>  x <- rcopula(gumbel.copula(1.4), 500)
>  ESTC <- fit.copula(empirical.copula(x), "gumbel")
>  ```

fitted.FARIMA Extract Fitted Values from a Fractional ARIMA Model

> `fitted.FARIMA(object, ...)`
>
> This is a method for the function `fitted.values()` for objects inheriting from class `"FARIMA"`. See `fitted.values` or `fitted.default` for the general behavior of this function and for the interpretation of `object`.

fitted.OLS Extract Fitted Values from Ordinary Least Squares

> `fitted.OLS(object, ...)`
>
> This is a method for the function `fitted.values()` for objects inheriting from class `"OLS"`. See `fitted.values` or `fitted.default` for the general behavior of this function and for the interpretation of `object`.

fitted.SUR Extract Fitted Values from a SUR Object

DESCRIPTION

This is a method for the function `fitted()` for objects inheriting from class `"SUR"`. See `fitted` or `fitted.default` for the general behavior of this function and for the interpretation of `object`.

```
fitted.SUR(object, ...)
```

VALUE

the matrix of fitted values of the SUR model.

fitted.unitroot Extract Fitted Values from Unit Root Regression

```
fitted.unitroot(object, ...)
```

This is a method for the function `fitted.values()` for objects inheriting from class `"unitroot"`. See `fitted.values` or `fitted.default` for the general behavior of this function and for the interpretation of `object`.

fitted.VAR Extract Fitted Values from a VAR Object

DESCRIPTION

This is a method for the function `fitted()` for objects inheriting from class `"VAR"`. See `fitted` or `fitted.default` for the general behavior of this function and for the interpretation of `object`.

```
fitted.VAR(object, ...)
```

VALUE

the matrix of fitted values of the VAR model. Note that the first `p` rows in the original series are deleted due to the lags, where `p` is the autoregressive order.

folio.dat Portfolio of Weekly Stock Returns

SUMMARY

This is a weekly `"timeSeries"` object with dimension `182 x 1618`, which runs from January 8, 1997 to June 28, 2000 and represents the stock returns on 1618 U.S. stocks.

forecast.object   Model Forecast Objects

DESCRIPTION
>   These are objects of class `"forecast"` that represent the predictions
>   or forecasts from a statistical model.

GENERATION
>   This class of objects is usually returned from the `predict`, or `cpredict`
>   function.

METHODS
>   The `"forecast"` class of objects currently has methods for the follow-
>   ing generic functions:
>
>   `plot`, `print`, `summary`.

STRUCTURE
>   The following components must be present in a legitimate `"forecast"`
>   object:

`values` : a vector, or a matrix that represents the predictions or forecasted
values.

`std.err` : a vector, or a matrix that represents the standard errors of the
predictions. It can be `NULL` if the standard errors were not computed.

`draws` : a matrix or an array of simulated forecasts, if simulation-based meth-
ods were used for prediction.

SEE ALSO
>   `cpredict`, `predict` , `plot.forecast` .

fplot Plot of Factor Returns/Scores

DESCRIPTION
Generates a trellis multi-panel object representing the factor returns
or factor scores.

```
fplot(object, index, hgrid=F, vgrid=F, main="Factor Returns", ...)
```

REQUIRED ARGUMENTS

object : a matrix, a data frame, or a `"timeSeries"` object with a matrix or
data frame in the data slot. This is usually generated by a call to the
generic `factors` function.

index : an integer vector, or a character vector giving the indices of the factors
to be returned. If `index` is a character vector, it must correspond to
the names of the factors.

OPTIONAL ARGUMENTS

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the trellis
plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the trellis plot.
The default is `FALSE`.

... : any other optional arguments that can be passed down to low level
plot functions.

VALUE
an object of class `trellis`, which is automatically plotted by `print.trellis`.
If this function is invoked directly without saving the returned object,
a trellis device will be open to draw the plot if one is not available yet.

SEE ALSO
`trellis.args`.

EXAMPLE
```
folio.f = mfactor(folio.dat, 5)
fplot(factors(folio.f))
```

frank.copula.object Frank Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"frank.copula"`, which inherits from `copula` and `archm.copula` classes.

GENERATION

This class of objects is constructed from the `frank.copula(delta)` function or `archm.copula` function.

METHODS

The `"frank.copula"` class of objects currently has methods for the following generic functions:

PHI, PhiDer, pcopula, dcopula, rcopula, contour.plot, LAMBDA<code>, <code>CondCinverse, Kendalls.tau, Spearmans.rho, dcdx.

Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula`, `contour.dcopula`, `contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"frank.copula"` class object:

parameters : value for the parameter `delta` (greater than 0).

param.names : `delta`.

param.lowbnd : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

param.upbnd : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

message : a message specifying the name of the parametric copula family (Frank copula family), and the copula class from which it inherits (Archimedean Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

Frank, M. J. (1979). On the simultaneous associativity of f(x, y) and x + y - f(x, y). *Aequationes Math.*, 19:194-229.

SEE ALSO

copula.object, archm.copula.object , archm.copula .

frip.dat Monthly Industrial Production of France

SUMMARY

This `"timeSeries"` vector contains the monthly industrial production
of France from January 1960 to December 1989.

galambos.copula.object   Galambos Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"galambos.copula"`, which inherits from `copula` and `ev.copula` classes.

GENERATION

This class of objects is constructed from the `galambos.copula(delta)` function or `ev.copula` function.

METHODS

The `"galambos.copula"` class of objects currently has methods for the following generic functions:

`Afunc, AfirstDer, AsecondDerPhiDer, Hderiv, pcopula, dcopula, rcopula, contour.plot, tail.index, Kendalls.tau, Spearmans.rho, dcdx.`

Furthermore, following functions are implemented for this class: `persp.dcopula, persp.pcopula, contour.dcopula, contour.pcopula.`

STRUCTURE

The following slots must be present in a legitimate `"galambos.copula"` class object:

`parameters` : a numerical value for the parameter `delta` (greater or equal to 0).

`param.names` : `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message`: a message specifying the name of the parametric copula family (Galambos copula family), and the copula class from which it inherits (Extreme Value Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

Galambos, J. (1975). Order statistics of samples from multivariate distributions. *Journal of the American Statistical Association*, 70:674-680.

SEE ALSO

`copula.object, ev.copula.object, ev.copula` .

garch.eq Function to Interpret Conditional Variance Specification

DESCRIPTION
> Translates `garch` variance equation.

> `garch.eq(x, leverage=F)`

REQUIRED ARGUMENTS

> x: one of the following specifications of a conditional variance structure in a GARCH model: '∼garch(p,q)', '∼egarch(p,q)', '∼pgarch(p,q)', '∼pgarch(p,q,d)', '∼tgarch(p,q)', '∼garch.2comp', '∼pgarch.2comp(d)', '∼egarch.2comp', '∼two.comp(type, d)'. Exogenous variables are also allowed.

leverage: logical variable. `TRUE` indicates to include the leverage effects.

VALUE
> a model list specifying the particular conditional variance structure.

SEE ALSO
> `fgarch.eq`, `mgarch.eq` , `mean.eq` .

garch.likelihood Log-Likelihood Function for GARCH/MGARCH Models

DESCRIPTION

Calculates the log-likelihood value for GARCH/MGARCH models. It
is not supposed to be called by the users directly.

```
garch.likelihood(PARPAR, ALLPAR=NULL, PINDEX=NULL, NINDEX=NULL,
    MLAG=NULL, NEQN=1, LEQN=1, NOBS=NULL, MEXO=1, VEXO=1, IVEC=NULL,
    y=NULL, x=NULL, z=NULL, EPS=NULL, H=NULL, CHTEPPFP=NULL, WG=NULL,
    AO1=NULL, BO1=NULL, CO1=NULL, AU=NULL, ABO=NULL, GALL=NULL,
    FI=NULL, LT=NULL, grd=T)
```

garch.model Create GARCH Model List

DESCRIPTION

Converts formula specifications into a model list that can be used by the functions `garch` and `simulate.garch` . Used primarily by `garch` and by users to create a model structure to edit to obtain particular model parameter values.

```
garch.model(formula.mean, formula.var, leverage=F, autoinit=T)
```

REQUIRED ARGUMENTS

`formula.mean`: a formula specifying the conditional mean equation structure of ARMA form. Can have the form y∼arma(p,q), where y is the response variable. The response variable y can be present or absent. If y does not appear in `formula.mean`, it must be specified by `series=y`. `arma(p,q)` can be replaced by `ar(p)`, `ma(q)`. A constant mean term is included by y∼1 or `1+arma(p,q)`, etc. Exogenous variables are also allowed in the formula.

`formula.var`: a formula specifying a GARCH model to fit, namely one of the following: '∼garch(p,q)', '∼pgarch(p,q)', '∼pgarch(p,q,d)', '∼tgarch(p,q)', '∼egarch(p,q)', or '∼two.comp(type,d)'. Exogenous variables are also allowed in the formula.

OPTIONAL ARGUMENTS

`leverage`: a logical flag: if `TRUE`, leverage effects are included in the model The default is `FALSE`.

`autoinit`: a logical flag: if `TRUE`, use the internal program to set initial values. These values may not be the best guess for a particular time series at hand.

VALUE

an object of class `"garch.model"`.

The conditional mean equation is specified by

`c.which` : a logical variable. `TRUE` indicates a constant is included in the conditional mean equation.

`c.value` : a scalar, the initial value for the constant term in the conditional mean equation.

AR: a list containing the following components: `order`, giving the order of the AR component; `which`, a logical vector of length equal to `order` indicating which coefficients are to be estimated; and `value`, a numerical vector giving the initial values of the AR coefficients.

MA: a list containing the following components: `order`, giving the order of the MA component; `which`, a logical vector of length equal to `order` indicating which coefficients are to be estimated; and `value`, a numerical vector giving the initial values of the MA coefficients.

X: a list containing the following components: `term`, giving the term labels for exogenous variables in the mean equation; and `value`, a numerical vector giving the initial values of the coefficients for exogenous variables.

The conditional variance equation is specified by

a.value: a scalar giving the initial value for the constant term in the variance equation.

arch: a list containing the following components: `order`, giving the order of the garch moving average; `which`, a logical vector of length `order` indicating which coefficients are to be estimated; `value`, a numeric vector giving the initial values of the coefficients; `lev.which`, a logical vector of length `order` indicating the lag steps with leverage effects; and `lev.value`, a numeric vector of length `order` giving the initial values of the leverage effects.

garch: a list containing the following components: `order`, giving the order of the GARCH autoregression; `which`, a logical vector of length `order` indicating the coefficients to be estimated; and `value`, a numeric vector of length `order` giving the initial values of the coefficients.

Z: a list containing the following components: `term`, giving the term labels for exogenous variables in the variance equation; and `value`, a numerical vector giving the initial values of the coefficients for exogenous variables.

power.which: logical, for `imod=3` (APARCH) only. This specifies the power GARCH model. `power.value` is the initial value of the power if `power.which=TRUE`. When `power.which=FALSE`, it specifies the power value.

Internal model names not needed by the users.

idmod: 0 indicating univariate model.

imod: integer of 1, 2, 3, 4, 5, 11, 12, 13, 14, 21, 22, 23, and 24 specifying univariate model types.

inpar: integer, 1 indicating to use the internal program to set default initial values. Otherwise, use the user supplied initial values.

SEE ALSO

`mgarch.model`, `fgarch.model` .

EXAMPLE

```
garch.ar5.mod = garch.model(y~ar(5), ~garch(1,1))
garch22.lev.mod = garch.model(~1, ~garch(2,2), leverage=T)
pgarch12.arma11.mod = garch.model(~arma(1,1), ~pgarch(1,2))
```

garch.stats Function to Compute Inference Statistics for GARCH Models

DESCRIPTION
> Computes major test statistics for GARCH models.

> `garch.stats(x, max.lag=NULL)`

REQUIRED ARGUMENTS

> x: an object of class `"garch"` and `"mgarch"`.

max.lag: the maximum lag steps in the Ljung-Box test.

VALUE
> The Shapiro and Wilk test, Jarque-Bera test for normality, Ljung-Box test, and Lagrange Multiplier test for serial correlations are computed.

SEE ALSO
> `summary.mgarch`.

garch Fit Univariate GARCH Models

DESCRIPTION

Fits one of a wide variety of univariate GARCH models, including not only the ARCH models of Engle (1982) and the basic GARCH models of Bollerslev (1986), but also the power GARCH (PGARCH), threshold GARCH (TGARCH), exponential GARCH (EGARCH), and the two component GARCH (TWO.COMP) models. In addition, this function allows the user to model ARCH-in-mean structure.

```
garch(formula.mean=~ arma(0,0), formula.var=~ garch(0,0),
    series=NULL, series.start=1, x=NULL, x.start=1, xlag=0,
    z=NULL, z.start=1, model=NULL, leverage=F, n.predict=1,
    trace=T, cond.dist="gaussian", dist.par=NULL, dist.est=T,
    control=NULL, algorithm="bhhh", ...)
```

REQUIRED ARGUMENTS

`formula.mean`: a formula specifying the conditional mean equation structure of ARMA form. Can have the form `y~arma(p,q)`, where `y` is the response variable. The response variable `y` can be present or absent. If `y` does not appear in `formula.mean`, it must be specified by the argument `series`. `arma(p,q)` can be replaced by `ar(p)`, `ma(q)`. A constant mean term is removed by `y~-1` or `-1+arma(p,q)`, etc. Exogenous variables are also allowed. There are three options available for the ARCH-in-mean structure: `sd.in.mean`, `var.in.mean`, and `logvar.in.mean`. To specify an `arma(1,1)` and a sd-in-mean structure, for example, the user only needs to use `y~arma(1,1)+sd.in.mean` for the `formula.mean` argument.

`formula.var`: a formula specifying a GARCH model to fit, namely one of the following: `~garch(p,q)`, `~pgarch(p,q)`, `~pgarch(p,q,d)`, `~tgarch(p,q)`, `~egarch(p,q)`, or `~two.comp(type,d)`.

`series`: name of the response variable.

`model`: a list specifying the model as returned by `garch.model`. The required arguments are `formula.mean` and `formula.var`, or `model`. If both `formula.mean` and `formula.var` are specified by the user, `garch` will call `garch.model` to generate a model list specification. If one but not both of `formula.mean` or `formula.var` is missing, the missing equation will be specified as the default: `arma(0,0)` for the conditional mean equation, and `garch(0,0)` for the conditional variance equation. If neither `formula.mean` nor `formula.var` is specified by the user, then a model list with the components specified the same way as in the model object created by `garch.model` must be supplied through

model argument, which has the default NULL. If the response variable y in the formula.mean is not specified, then it must be supplied in the argument series, which also has the default value NULL.

OPTIONAL ARGUMENTS

series.start: the element of series to begin with. For example, if series.start=10, the observations of series will start from the tenth element of series.

x: a vector or matrix of exogenous variables in the mean equation. Preferably this should be specified in formula.mean.

x.start: the element of x to begin with. For example, if x.start=10, the observations of x will start from the tenth row of x.

xlag: lag steps for exogenous variables in the mean equation.

z: a vector or matrix of exogenous variables in the variance equation. Preferably this should be specified in formula.var.

z.start: the element of z to begin with. For example, if z.start=10, the observations of z will start from the tenth row of z.

leverage: logical flag. TRUE indicates to include leverage terms. The default is FALSE.

n.predict: number of steps to predict.

cond.dist: conditional distribution given the past. The default is "gaussian". Other options are "t", "double.exp", and "ged" (generalized error distribution).

dist.par: the parameter for the conditional distribution. If the conditional distribution is "t", dist.par is the degrees of freedom. If the conditional distribution is "ged", dist.par is the power of the exponent.

dist.est: logical flag: if TRUE, the parameters of the conditional distribution will be estimated. The default is FALSE.

trace: logical flag: if TRUE, step-by-step optimization results will be printed on the screen, otherwise see opt.index for convergence information. The default is TRUE.

control: a list of control parameters to be used in the numerical algorithms. See bhhh.control for the possible control parameters and their default settings.

algorithm: parameter that determines the algorithm used for maximum likelihood estimation. Currently only "bhhh" is available.

VALUE

an S version 3 object of class `"garch"`, which contains the following components:

`series :` a numeric vector, or a `"timeSeries"` object, which represents the observed time series.

`call:` the image of the original call which generated the object.

`coef:` the estimated coefficients in the model.

`likelihood:` the log-likelihood value at the optimum.

`residuals:` a numeric vector, or a `"timeSeries"` object, which represents the residuals from the fitted model.

`sigma.t:` a numeric vector, or a `"timeSeries"` object, which represents the estimated conditional standard deviation sequence.

`df.residual:` the residual degree of freedom. That is, `df.residual` is equal to the number of observations minus the number of parameters in the estimated model.

`cond.dist:` a list with the following components giving the estimated error distribution: `cond.dist` which is a character string giving the name of the distribution; `dist.par` which gives the distribution parameter such as the degree of freedom if any; `dist.est` which is a logical flag indicating if `dist.par` was estimated by MLE.

`model:` a `"garch.model"` object which a model list created and returned by the function `garch.model`, or a model list supplied by the user as an input to `garch` function.

`cov:` a list of matrices representing the outer product of gradient and the numeric Hessian matrix at the optimum. This is used to compute the covariance matrix of the estimated parameters.

`asymp.sd:` the asymptotic or long run variance of the GARCH model, if the estimated GARCH model is stationary. This is `NA` if the estimated GARCH model is non-stationary.

`prediction:` a `"predict.garch"` object which represents the predictions of the estimated GARCH model.

`opt.index:` convergence indicator:

1 indicating BHHH convergence has been reached.

2 indicating the likelihood value has converged.

3 indicating a local maximum has been reached.

4 indicating the maximum number of iterations has been reached.

REFERENCES

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50:987-1006.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroscedasticity. *Journal of Econometrics*, 31:307-327.

SEE ALSO

`bhhh.control`, `fgarch` , `garch.model` , `mgarch` ,`predict.garch` .

EXAMPLE

```
# GARCH(1,1) model with constant conditional mean
# and conditional Gaussian error distribution
hp.s.mod = garch(hp.s~1, ~garch(1,1))

# GARCH(1,1) model with constant conditional mean
# and conditional t distribution
hp.s.mod.t = garch(hp.s~1, ~garch(1,1), cond.dist="t")

# PGARCH(1,1) model
hp.s.mod.p = garch(hp.s~1, ~pgarch(1,1))

# GARCH(1,2) model with conditional mean
# a constant plus an AR(2) term
hp.s.modar = garch(hp.s~ar(2), ~garch(1,2))
```

getReturns Financial Return Series

DESCRIPTION
    Computes the return series given a financial security price series.

        getReturns(x, type="continuous", percentage=F, trim=T)

REQUIRED ARGUMENTS

    x: a vector or a `"timeSeries"` object representing the price series.

OPTIONAL ARGUMENTS

    type: a character string specifying the type of returns to be computed. Valid
        choices are: `"continuous"` and `"discrete"`. If `type="continuous"`,
        the returns are calculated as the logarithm differences; if `type="discrete"`,
        the returns are calculated as percentage changes. The default is `"continuous"`

    percentage: a logical flag: if `TRUE`, the return series will be expressed in per-
        centage points. The default is `FALSE`.

    trim: a logical flag: if `TRUE`, the first missing observation in the return series
        will be removed. The default is `TRUE`.

VALUE
    a vector or a `"timeSeries"` object representing the return series.

DETAILS
    If there are missing values in the price series, they will generate missing
    values in the return series.

EXAMPLE
    sp500.ret = getReturns(nelson.dat[21:129, "SP500"])

GetSsfArma State Space Form of an ARMA Model

DESCRIPTION

Returns the minimal necessary components for a state space form of a univariate autoregressive and moving average model with or without regressors.

```
GetSsfArma(ar=NULL, ma=NULL, sigma=1, model=NULL)
GetSsfRegArma(mX, ar=NULL, ma=NULL, sigma=1, model=NULL)
```

REQUIRED ARGUMENTS

mX : a rectangular numeric object which represents the regressors in the model. Note that variables must be in columns.

OPTIONAL ARGUMENTS

ar : a numeric vector giving the autoregressive coefficients.

ma : a numeric vector giving the moving average coefficients.

sigma : the standard deviation of the disturbance term. The default is 1.

model : a list which contains the ar, ma, and sigma components of an ARMA model. If model is given, then ar, ma and sigma is ignored.

VALUE

a list which contains the following components:

mPhi : an (m+cX+1) x (m+cX) matrix that gives the coefficient matrix of the state variables, where cX is the number of regressors and m=max(p, q+1) with p being the autoregressive order and q being the moving average order.

mJPhi : an (m+cX+1) x (m+cX) matrix that determines the specification of regressors. This is only returned by GetSsfRegArma function. See the help file for CheckSsf for details.

mOmega : an (m+cX+1) x (m+cX+1) matrix that gives the variance-covariance matrix of the disturbance of the state space form.

mSigma : a (m+1) x m matrix giving the asymptotic mean and variance of the ARMA model. See the help file for CheckSsf for details.

mX : the same as the input mX object except that it is coerced as a "matrix" object. This component is only returned by GetSsfRegArma function.

DETAILS

The ARMA model is assumed to be of the form:

```
y(t) = phi(1)y(t-1) + ...  + phi(p)y(t-p) + e(t) + theta(1)e(t-1)
+ ...  + theta(q)e(t-q)
```

Note that in this form the signs of the moving average coefficients are the opposite of those required by `arima.mle` function.

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

SEE ALSO

`CheckSsf`,`GetSsfReg` ,`arima.mle` .

EXAMPLE

```
# ARMA(2, 2) model
GetSsfArma(ar=c(0.5, 0.2), ma=c(0.1, 0.1))

# AR(2) model
ar.mod = list(ar=c(0.5, 0.2), sigma=0.5)
GetSsfArma(model=ar.mod)

# AR(2) with constant
GetSsfRegArma(rep(1,100), ar=c(0.5, 0.2))
```

GetSsfReg State Space Form of a Linear Regression Model

DESCRIPTION

Returns the minimal necessary components for a state space form of a linear regression model.

```
GetSsfReg(mX)
```

REQUIRED ARGUMENTS

mX : a rectangular numeric object which represents the regressors in the model. Note that variables must be in columns.

VALUE

a list which contains the following components:

mPhi : a `(cX+1) x cX` matrix that gives the coefficient matrix for the state variables, where `cX` is the number of regressors in `mX`.

mJPhi : a `(cX+1) x cX` matrix that determines the specification of the regressors. See the help file for **CheckSsf** for details.

mOmega : a `(cX+1) x (cX+1)` matrix that gives the variance-covariance matrix of the disturbance of the state space model

mSigma : a `(cX+1) x cX` matrix that gives the diffuse initial conditions of the state variables. See the help file for **CheckSsf** for details.

mX : the same as the input `mX` object except that it is coerced as a `"matrix"` object.

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

SEE ALSO

CheckSsf,GetSsfRegArma .

EXAMPLE

```
# a linear time trend model
GetSsfReg(cbind(1, 1:100))
```

GetSsfSpline State Space Form of a Cubic Spline Model

DESCRIPTION
>   Returns the minimal necessary components for a state space form of
>   a cubic spline model.

>   `GetSsfSpline(snr=1, delta=0)`

OPTIONAL ARGUMENTS

  `snr` : the signal-to-noise ratio. The default is 1.

  `delta` : a numeric vector that gives the gaps between observations.

  VALUE
>   a list which contains the following components:

  `mPhi` : a 3 x 2 matrix that gives the coefficient matrix of the state variables.

  `mJPhi` : a 3 x 2 matrix that determines the time varying aspect of `mPhi`.

  `mOmega` : a 3 x 3 matrix that gives the covariance matrix of the disturbance
>   of the state space form.

  `mJOmega` : a 3 x 3 matrix that determines the time varying aspect of `mOmega`.

  `mSigma` : a 3 x 2 matrix that gives the diffuse initial conditions of the state
>   variables. See the help file for `CheckSsf` for details.

  `mX` : a `cT` x 4 matrix that gives the time varying variables of the state
>   space form, where `cT` is the length of `delta`.

REFERENCES
>   Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State
>   Space Methods*. Oxford University Press.

>   Koopman, S. J., Shephard, N., and Doornik, J. A. (1999). Statis-
>   tical algorithms for methods in state space form using SsfPack 2.2.
>   *Econometric Journal*, 2:113-166.

SEE ALSO
>   `CheckSsf`,`hpfilter` .

EXAMPLE
>   `GetSsfSpline(snr=0.2, delta=1:5)`

GetSsfStsm State Space Form of Unobserved Components Time Series
Models

DESCRIPTION

Returns the state space form of an unobserved components time series
model, or structural time series model, or dynamic linear model.

```
GetSsfStsm(irregular=1, level=0.1, slope=NULL,
    seasonalDummy=NULL, seasonalTrig=NULL, seasonalHS=NULL,
    cycle0=NULL, cycle1=NULL, cycle2=NULL, cycle3=NULL,
    cycle4=NULL, cycle5=NULL, cycle6=NULL, cycle7=NULL,
    cycle8=NULL, cycle9=NULL)
```

OPTIONAL ARGUMENTS

`irregular` : the standard deviation of the irregular term. The default is set
to 1.

`level` : the standard deviation of the disturbance term in the level equation.
The default is set to 0.1.

`slope` : the standard deviation of the disturbance term in the slope equation.
If NULL, then the slope term is not included. The default is NULL.

`seasonalDummy` : a vector with two elements, with the first being the standard
deviation of the sum of the seasonal dummies, and the second being
the number of seasonal dummies. If NULL, then seasonal dummy terms
are not included. The default is NULL.

`seasonalTrig` : a vector with two elements, with the first being the standard
deviation of the sum of the trigonometric seasonal terms, and the sec-
ond being the number of seasonal terms. If NULL, then trigonometric
seasonal terms are not included. The default is NULL.

`seasonalHS` : a vector with two elements, with the first being the standard
deviation of the sum of the Harrison-Stevens seasonal terms, and the
second being the number of seasonal terms. If NULL, then Harrison-
Stevens seasonal terms are not included. The default is NULL.

`cycle0`, ..., `cycle9` : a vector with three elements, with the first being the
standard deviation of the initial condition, the second the frequency of
the cycle, and the third the damping factor. If NULL, the cycle terms
are not included. The default is NULL.

VALUE

a list which contains the following components:

mPhi: a numeric matrix that gives the coefficient matrix of the state variables in the state space form.

mOmega: a numeric matrix that gives the covariance matrix of the disturbance of the state space form.

mSigma: a numeric matrix that gives the initial distribution of the state variables.

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

SEE ALSO

CheckSsf,stl .

EXAMPLE

```
# a local level model
GetSsfStsm(irregular=1, level=0.5)
```

gev.lmom L-Moment Parameter Estimates of GEV Distribution

DESCRIPTION
>   Computes L-Moment parameter estimates for GEV distribution.

>   `gev(lmom)`

REQUIRED ARGUMENTS

`lmom` : vector of estimates of sample L-Moments in this order: l1, l2, t3.
Missing values (NAs) are not allowed.

VALUE
>   an object of class `"gev"` representing the result. See `gev.object` for
details.

DETAILS
>   The argument of the function should be a vector of L-Moments. The
function `sample.LMOM` computes unbiased estimates of sample L-Moments,
but other estimates can also be used for that purpose (for example,
plotting position estimates). If the argument of the function is a vector
with length greater than four, the function will assume that a sample
is passed as a first argument. `sample.LMOM` is used to calculate the
L-Moments estimates and a warning message will be printed to the
standard output.

>   The L-Moments method for parameter estimation consists of equat-
ing estimates of sample L-Moments to a distribution's L-Moment, ex-
pressed in terms of the distribution's parameters.

>   `gev.lmom` is named `estimate.lmom.gev` in the original EVANESCE
library.

REFERENCES
>   Hosking, J. R. M., and Wallis, J. R. (1987). Parameter and quan-
tile estimation for the generalized pareto distribution. *Technometrics*,
29(3):339-349.

>   Hosking, J. R. M. (1986). The theory of probability weighted mo-
ments. Research Report RC12210, IBM Research, Yorktown Heights,
NY.

>   Hosking, J. R. M. (1990). L-moments: analysis and estimation of
distributions using linear combinations of order statistics. *Journal of
Royal Statistical Society*, 52(1):105-124.

>   Hosking, J. R. M., Wallis, J. R., and Wood, E. F. (1985). Estima-
tion of the generalized extreme value distribution by the method of
probability-weighted moments. *Technometrics*, 27(3):251-261.

SEE ALSO

gev.object, gev ,gev.mix1 ,gev.mix2 .

EXAMPLE

```
x = rgev(500, sigma = 3.5, mu=0, xi = 0.4)
lmomx = sample.LMOM(x)
gev.lmom(lmomx)
```

gev.mix1   MIX1 and MIX2 Parameter Estimates of GEV Distribution

DESCRIPTION
Computes MIX1 and MIX2 parameter estimates for GEV distribution.

```
gev.mix1(sample, init.est=NA, sampLmom=NA, epsilon=1e-5)
gev.mix2(sample, init.est=NA, sampLmom=NA, epsilon=1e-5)
```

REQUIRED ARGUMENTS

`sample` : a numeric vector of data points or `"timeSeries"` object. Missing values (NAs) are not allowed.

OPTIONAL ARGUMENTS

`init.est` : If a vector of parameter estimates obtained by methods other than MIX1 or MIX2 is available, it may be passed into the function. It will be used as an initialization point for the optimization routine in the MIX1 or MIX2. If no argument is specified, L-Moment estimate will be computed and used to initialize the MIX1 or MIX2 routines.

`epsilon` : the relative tolerance of the optimization routine for MIX1 or MIX2. The default value is `10e-5`.

VALUE
an object of class `"gev"` representing the result. See `gev.object` for details.

DETAILS
Splus standard optimization `nlmin` function is used to find the conditional maximum of the log-likelihood function. If `nlmin` does not converge, a warning is issued.

In the MIX1 method, the log-likelihood function `L(sigma, mu, k|x)` is maximized as a function of `sigma` and `k`, after substituting `mu` from the L-Moment equation; In MIX2, the likelihood function `L(sigma, mu, k|x)` is maximized as a function of `k` after using `mu` and `sigma` from the L-Moment equations method.

`gev.mix1` and `gev.mix2` are named `estimate.mix1.gev` and `estimate.mix2.gev` in the original EVANESCE library.

REFERENCES
Morrison, J. E. (2001). *Extreme Value Statistics with Applications in Hydrology and Financial Engineering.* Ph.D. thesis, Princeton University.

SEE ALSO
`gev.object`, `gev` ,`gev.lmom` .

EXAMPLE

```
x <- rgev(500, sigma = 3.5, mu=0, xi = 0.4)
gev.mix1(x)
```

gev.object GEV (Generalized Extreme Value Distribution) Model Object

DESCRIPTION

These are objects of class `"gev"`. They represent the fit of a GEV model to block maxima data.

GENERATION

This class of objects is returned from the `gev` function.

METHODS

The `"gev"` class of objects currently has methods for the following generic functions:

`plot`.

STRUCTURE

The following components must be present in a legitimate `"gev"` object:

`n.all` : the number of observation in the raw data if data is supplied as raw data. Otherwise `n.all` is `NA`.

`n` : the number of blocks.

`call` : the image of the original call that generated the object.

`data` : the block maxima data (supplied or calculated).

`block` : the block size as supplied.

`par.ests` : the parameter estimates.

`par.ses` : the standard error of parameter estimates.

`varcov` : the variance-covariance matrix of parameter estimates.

`converged` : the logical flag indicating if ML estimation has converged.

`nllh.final` : the final negative loglikelihood function value. It is `NA` if `gev.lmom` is used.

SEE ALSO

`gev`.`gev.lmom`.`gev.mix1`.`gev.mix2` .

gev Fit Generalized Extreme Value Distribution

DESCRIPTION

Fits a generalized extreme value distribution (GEV) to block maxima data.

```
gev(data, block=NA, max.fcal=500, max.iter=200)
```

REQUIRED ARGUMENTS

data : a numeric vector or `"timeSeries"` object. The interpretation of `data` depends on the value of `block`: if no block size is specified then `data` is interpreted as block maxima; if block size is set, then `data` is interpreted as raw data and block maxima are calculated internally.

OPTIONAL ARGUMENTS

block : an integer specifying the block size, or a character string that takes the value of `"month"`, `"quarter"`, `"semester"` or `"year"` (if `data` is a `"timeSeries"` object with calendar times), or simply `NA` when block maxima data are supplied. The default is `NA`.

max.fcal : an integer specifying the maximum number of function evaluations allowed in maximum likelihood estimation. The default is 500.

max.iter : an integer specifying the maximum number of iterations allowed in maximum likelihood estimation. The default is 200.

VALUE

an object of class `"gev"` representing the fit. See `gev.object` for details.

SEE ALSO

plot.gev, gumbel ,gev.object . gev.lmom .gev.mix1 .gev.mix2 .

EXAMPLE

```
# Fit GEV to monthly maxima of daily returns on BMW share price
out = gev(bmw, "month")

# Fit GEV to maxima of blocks of 100 observations
out = gev(bmw, 100)

# Fit GEV to the data in nidd.annual
out = gev(nidd.annual)
```

gpd.1p Cumulative Probability and Quantiles of a Fitted GPD Object

DESCRIPTION

Calculates the cumulative probability and quantiles of a fitted GPD object using semi-parametric estimation methods.

```
gpd.1p(x, obj, linear=T)
gpd.2p(x, est.object, linear=T)
gpd.1q(p, obj, linear=T)
gpd.2q(p, est.object, linear=T)
```

REQUIRED ARGUMENTS

x : a numeric vector or `"timeSeries"` of quantiles.

p : a numeric vector of probabilities.

obj : a `"gpd"` object.

est.object : a `"gpd"` object.

OPTIONAL ARGUMENTS

linear : a logical value specifying if the empirical CDF and quantile function are linearly interpolated for `lower < x < upper`. If `FALSE`, the empirical CDF and quantile function are returned for `lower < x < upper`. The default is `TRUE`.

VALUE

The returned values of functions `gpd.1p` and `gpd.2p` is a vector of the same length as x comprising the estimates computed at the points x of the cumulative distribution function from which the sample data was issued.

The returned value of functions `gpd.1q` and `gpd.2q` is an estimate of the quantile function of a random variable, represented by the data, at points p. It is a vector of the same length as p.

DETAILS

`gpd.1p` is named `cdf.estimate.1tail`, `gpd.2p` is named `cdf.estimate.2tails`, `gpd.1q` is named `quant.estimate.1tail`, `gpd.2q` is named `quant.estimate.2tails` in the original EVANESCE library.

REFERENCES

Prescott, P., and Walden, A. T. (1980). Maximum likelihood estimation of the parameters of the generalized extreme-value distribution. *Biometrika*, 67(3):723-724.

Embrechts, P., Kluppelberg, C., and Mikosch, T. (1997). *Modelling Extremal Events for Insurance and Finance.* New York: Springer-Verlag.

SEE ALSO

    gpd.lmom, gpd.ml ,gpd.object . gpd .

EXAMPLE

```
tmp.dat = BRAZIL.COFFEE[,2][as.matrix(seriesData(BRAZIL.COFFEE[,2])) !=0]
tmp = gpd.tail(tmp.dat, upper=0.015, lower=-0.015,
    upper.method="lmom", lower.method="lmom", plot=F)
gpd.1p(x=0.06, obj=tmp)
```

gpdbiv.object   Bivariate POT (peaks over thresholds) Model Object

DESCRIPTION

These are objects of class `"gpdbiv"`. They represent the fit of a bivariate POT (peak over thresholds) model for joint excesses over thresholds.

GENERATION

This class of objects is returned from the `gpdbiv` function.

METHODS

The `"gpdbiv"` class of objects currently has has methods for the following generic functions:

`plot`.

STRUCTURE

The following components must be present in a legitimate `"gpdbiv"` object:

`data1` : the exceedances data of the first dataset (over the specified threshold `u1`).

`delta1` : a logical vector indicating if they are exceedances of the second dataset at exceedances data points for the first dataset.

`data2` : the exceedances data of the second dataset (over the specified threshold `u2`).

`delta2` : a logical vector indicating if they are exceedances of the first dataset at exceedances data points for the second dataset.

`u1` : the threshold for the first dataset.

`ne1` : the number of exceedances for the first dataset.

`lambda1` : the proportion of the first dataset less than the threshold `u1`.

`u2` : the threshold for the second dataset.

`ne2` : the number of exceedances for the second dataset.

`lambda2` : the proportion of the second dataset less than the threshold `u2`.

`alpha` : the correlation coefficient estimate.

`alpha.se` : the standard error of the correlation coefficient estimate.

`par.ests1` : the parameter estimates vector of a marginal excess distribution (GPD fit) for the first dataset.

`par.ses1` : the standard errors vector of `par.ests1`.

`par.ests2` : the parameter estimates vector of a marginal excess distribution (GPD fit) for the second dataset.

`par.ses2` : the standard errors vector of `par.ests2`.

`converged` : a logical flag indicating if ML estimation converged.

`nllh.final` : the negative maximum log-likelihood function value.

`Dependence` : the type of the dependence structure.

`dep.func` : the mathematical expression of the dependence structure.

SEE ALSO

`gpdbiv`.

gpdbiv Bivariate POT Model

DESCRIPTION

Fits a bivariate POT (peaks over thresholds) model for joint excesses over thresholds.

```
gpdbiv(data1= NA, data2= NA, u1=NA, u2=NA, ne1=NA, ne2=NA,
       global=F)
```

REQUIRED ARGUMENTS

data1: a numeric vector or `"timeSeries"` object.

data2: a numeric vector or `"timeSeries"` object.

u1: a numeric value of threshold for `data1`. Either `u1` or `ne1` must be given but not both. The default is `NA`.

u2: a numeric value of threshold for `data2`. Either `u2` or `ne2` must be given but not both. The default is `NA`.

ne1: a number of upper extremes to be used for `data1`. Either `ne1` or `u1` must be given but not both. The default is `NA`.

ne2: a number of upper extremes to be used for `data2`. Either `ne2` or `u2` must be given but not both. The default is `NA`.

OPTIONAL ARGUMENTS

global: a logical flag: if `TRUE`, a global maximization of the likelihood is undertaken with respect to marginal and dependence parameters. A two-stage local fit is undertaken if `FLASE`, where first the marginal parameters are estimated and then the dependence parameter. A local fit is faster than a global fit. The default is `FALSE`.

VALUE

an object of class `"gpdbiv"` representing the fit. See `gpdbiv.object` for details.

DETAILS

The function implements a model suggested by Richard Smith. (see references below). The marginal excess distributions are GPD distributions, as suggested by univariate EVT and implemented in `gpd` function. The dependence specification is known as the logistic or Gumbel dependence structure, but it would be easy to program alternatives.

REFERENCES

Smith, R. L. (1994). Multivariate threshold methods. in J. Galambos (ed.) *Extreme Value Theory and Applications*. Kluwer.

Smith, R. L., Tawn, J. A., and Coles, S. G. (1997). Markov chain models for threshold exceedances. *Biometrika*, 84:249-268.

SEE ALSO

gpd, interpret.gpdbiv, plot.gpdbiv, gpdbiv.object,fit.copula
.

EXAMPLE

```
out = gpdbiv(-bmw, -siemens, ne1=100, ne2=100)
interpret.gpdbiv(out)
plot(out)
```

gpdjoint.1p Bivariate Joint Cumulative Probability Function

DESCRIPTION
>   Calculates the bivariate joint cumulative probability based on copula
>   and fitted GPD objects using semi-parametric estimation methods.

>       ```
>       gpdjoint.1p(x, y, copula, x.est, y.est)
>       gpdjoint.2p(x, y, copula, x.est, y.est)
>       ```

REQUIRED ARGUMENTS

>   x: a numeric vector or `"timeSeries"` of quantiles.

>   y: a numeric vector or `"timeSeries"` of quantiles.

>   copula: a `"empirical.copula"` object, or a parametric `copula` fitted to the
>   data.

>   x.est: a `"gpd"` object.

>   y.est: a `"gpd"` object.

VALUE
>   One-tail or two-tail cumulative probabilities for given pairs of quan-
>   tiles. It is a vector of the same length as x.

DETAILS
>   `gpd.1p` is named `jointcdf.est.1tail`, and `gpd.2p` is named `jointcdf.est.2tails`
>   in the original EVANESCE library.

SEE ALSO
>   `gpd.1p`, `gpd.2p` .

EXAMPLE
>       ```
>       tmp = gpd.tail(UTI.ELEC[,2], upper=0.005, lower=-0.005, plot=F)
>       tmp1 = gpd.tail(UTI.GAS[,2], upper=0.005, lower=-0.005, plot=F)
>       gpdjoint.2p(c(-0.015,0.015, Inf, 0.015), c(-0.015, 0.015, 0.015, Inf),
>           empirical.copula(gpd.2p(UTI.ELEC[,2], tmp), gpd.2p(UTI.GAS[,2], tmp1)),
>           x.est=tmp, y.est=tmp1)
>       ```

gpd.lmom L-Moment Parameter Estimates for GPD Distribution

DESCRIPTION

Computes L-Moment parameter estimates for GPD distribution.

`gpd.lmom(lmom, sample=NA, location=NA)`

REQUIRED ARGUMENTS

`lmom` : a vector of estimates of sample L-Moments in this order: l1, l2, t3. Missing values (NAs) are not allowed.

`location`: If the value of the GPD location parameter `mu` is known, it may be specified here. Either `locaiton` or `sample` should be specified.

`sample`: a numeric vector of data points or `"timeSeries"` object. Either `locaiton` or `sample` should be specified.

VALUE

a list containing the following components:

`n` : the number of sample data (if supplied).

`data` : the original sample data if supplied.

`par.ests` : parameter estimates of `beta`, `mu`, and `xi` (or `k`).

DETAILS

`gpd.lmom` is named `estimate.lmom.gpd` in the original EVANESCE library.

REFERENCES

Hosking, J. R. M., and Wallis, J. R. (1987). Parameter and quantile estimation for the generalized pareto distribution. *Technometrics*, 29(3):339-349.

Hosking, J. R. M. (1986). The theory of probability weighted moments, Research Report RC12210, IBM Research, Yorktown Heights, NY.

Hosking, J. R. M. (1990). L-moments: analysis and estimation of distributions using linear combinations of order statistics. *Journal of Royal Statistical Society*, 52(1):105-124.

Hosking, J. R. M., Wallis, J. R., and Wood, E. F. (1985). Estimation of the generalized extreme value distribution by the method of probability-weighted moments. *Technometrics*, 27(3):251-261.

SEE ALSO

`gpd.tail`, `gpd.ml` .

EXAMPLE

```
gpd.lmom(sample=BRAZIL.COFFEE[,2])
```

gpd.ml ML Parameter Estimates of GPD Distribution

DESCRIPTION

Computes ML parameter estimates for GPD distribution.

`gpd.ml(sample, location=NA, init.est=NA, epsilon=1e-005)`

REQUIRED ARGUMENTS

`sample` : a numeric vector of data points or `"timeSeries"` object. Missing values (NAs) are not allowed.

OPTIONAL ARGUMENTS

`location`: If the value for GPD location parameter `mu` is known, it may be specified here. The ML method is, then, to estimate the other two parameters.

`init.est`: If a vector of parameter estimates obtained by methods other than method of maximum likelihood is available, it may be passed into the function. It will be used as an initialization point for the optimization routine in the ML. If no argument is specified, L-Moment estimates will be computed and used to initialize the ML.

`epsilon`: the relative tolerance of the optimization routine in the MLE. The default value is `10e-5`.

VALUE

a list containing the following components:

`n` : the number of sample data.

`data` : the original sample data.

`par.ests` : parameter estimates of `beta`, `mu`, and `xi` (or k).

`convergence` : the convergence status in the ML estimation.

`nllh.final` : the final negative log-likelihood function value.

DETAILS

`gpd.ml` is named `estimate.mle.gpd` in the original EVANESCE library.

REFERENCES

Prescott, P., and Walden, A. T. (1980). Maximum likelihood estimation of the parameters of the generalized extreme-value distribution. *Biometrika*, 67(3):723-72.

Embrechts, P., Kluppelberg, C., and Mikosch, T. (1997). *Modelling Extremal Events for Insurance and Finance.* New York: Springer-Verlag.

SEE ALSO

`gpd.tail`, `gpd.lmom` .

EXAMPLE

`gpd.ml(BRAZIL.COFFEE[,2][as.matrix(seriesData(BRAZIL.COFFEE[,2])) > 0])`

gpd.object GPD (Generalized Pareto Distribution) Model Object

DESCRIPTION

These are S version 3 objects of class `"gpd"`. They represent the fit of a GPD model for excesses over a high threshold.

GENERATION

This class of objects is returned from the `gpd` function.

METHODS

The `"gpd"` class of objects currently has methods for the following generic functions:

`plot`.

STRUCTURE

The following components must be present in a legitimate `"gpd"` object:

`n` : the length of `data`.

`data` : the input data (sorted).

`upper.exceed` : the exceedances data at the upper tail.

`lower.exceed` : the exceedances data at the lower tail.

`upper.thresh` : the upper threshold.

`lower.thresh` : the lower threshold.

`p.less.upper.thresh` : the percentage of the data less than the upper threshold.

`p.larger.lower.thresh` : the percentage of the data larger than the lower threshold.

`n.upper.exceed` : the length of exceedances at the upper tail.

`n.lower.exceed` : the length of exceedances at the lower tail.

`upper.method` : the method of estimation for upper tail.

`lower.method` : the method of estimation for lower tail.

`upper.par.ests` : the parameter estimates vector for upper tail.

`lower.par.ests` : the parameter estimates vector for lower tail.

`upper.par.ses` : the standard error of parameter estimates for upper tail.

`lower.par.ses` : the standard error of parameter estimates for lower tail.

`upper.varcov` : the variance-covariance matrix of parameter estimates for upper tail.

`lower.varcov` : the variance-covariance matrix of parameter estimates for lower tail.

`upper.info` : the method of calculating `varcov` for upper tail.

`lower.info` : the method of calculating `varcov` for upper tail.

`upper.converged` : the logical flag indicating if ML estimation converged for upper tail. It is `NA` if `method="pwm"` or 2-tail estimation is performed using `gpd.tail` function.

`lower.converged` : the logical flag indicating if ML estimation converged for lower tail.

`upper.nllh.final` : the negative maximum log-likelihood function value if `method="ml"` is used in upper tail. It is `NA` if other `method` is used.

`lower.nllh.final` : the negative maximum log-likelihood function value if `method="ml"` is used in lower tail. It is `NA` if other `method` is used.

SEE ALSO

`gpd.gpd.tail` .

gpd.q Calculate Quantile Estimates and Add to a GPD plot

DESCRIPTION

Calculates quantile estimates and confidence intervals for high quantiles above the threshold in a GPD analysis and add them to the current GPD plot (such as `plot.gpd` or `tailplot`).

```
gpd.q(pp, ci.type="likelihood", ci.p=0.95, like.num=50,
      plot=F)
```

REQUIRED ARGUMENTS

pp: a number between 0 and 1 specifying the desired probability for quantile estimate (e.g. `0.99` for the 99th percentile).

OPTIONAL ARGUMENTS

ci.type: a character string specifying method for calculating a confidence interval. It takes value of either `"likelihood"` or `"wald"`. The default is `"likelihood"`.

ci.p: a number between 0 and 1 specifying the probability for confidence interval (must be less than `0.999`). The default is `0.95`.

like.num: an integer specifying the number of times to evaluate profile likelihood. The default is `50`.

plot: a logical flag: if `TRUE`, the quantile estimate and its confidence interval are added onto the current GPD plot. The default is `FALSE`.

VALUE

a vector containing the lower confidence interval bound (`"Lower CI"`), quantile estimate (`"Estimate"`), standard error of the quantile estimate (`"Std.Err"`) if ci.type is set to `"wald"`, and the upper confidence interval bound (`"Upper CI"`). The quantile estimate and its confidence intervals are added onto the current GPD plot if `plot=T`.

DETAILS

The GPD approximation in the tail is used to estimate quantile. The `"wald"` method uses the observed Fisher information matrix to calculate confidence interval. The `"likelihood"` method reparametrizes the likelihood in terms of the unknown quantile and uses profile likelihood arguments to construct a confidence interval.

SEE ALSO

gpd, `plot.gpd` , `gpd.sfall` .

EXAMPLE

```
out = gpd(danish,10)
tailplot(out)
# Estimates 99.9th percentile of Danish fire losses
gpd.q(0.999, plot=T)
```

gpd.sfall Calculate the Expected Shortfall Estimates and Add to a GPD plot

DESCRIPTION

Calculates expected shortfall (tail conditional expectation) estimates and confidence intervals for high quantiles above the threshold in a GPD analysis and add them to the current GPD plot (such as `plot.gpd` or `tailplot`).

```
gpd.sfall(pp, ci.p=0.95, like.num=50, plot=F)
```

REQUIRED ARGUMENTS

pp: a number between 0 and 1 specifying the desired probability for expected shortfall estimate (e.g. `0.99` for the 99th percentile).

OPTIONAL ARGUMENTS

ci.p: a number between 0 and 1 specifying the probability for confidence interval (must be less than `0.999`). The default if `0.95`.

like.num: an integer specifying the number of times to evaluate profile likelihood. The default if `50`.

plot: a logical flag: if `TRUE`, the expected shortfall estimate and its confidence interval are added onto the current GPD plot. The default is `FALSE`.

VALUE

a vector containing the lower confidence interval bound (`"Lower CI"`), quantile estimate (`"Estimate"`), and the upper confidence interval bound (`"Upper CI"`). The expected shortfall estimate and its confidence intervals (the profile likelihood curve) are added onto the current GPD plot if `plot=T`.

DETAILS

Expected shortfall is the expected size of the loss, given that a particular quantile of the loss distribution is exceeded. The GPD approximation in the tail is used to estimate expected shortfall. The likelihood is reparametrised in terms of the unknown expected shortfall and profile likelihood arguments are used to construct a confidence interval.

SEE ALSO

`gpd`, `plot.gpd` , `tailplot` , `gpd.q` .

EXAMPLE

```
out = gpd(danish,10)
tailplot(out)
# Estimates associated expected shortfall for Danish fire losses
gpd.sfall(0.999, plot=T)
```

gpd.tail Fit Generalized Pareto Distribution using Peaks over Threshold
Method

DESCRIPTION

Fits a generalized Pareto distribution (GPD) to exceedances data over
certain thresholds on two tails.

```
gpd.tail(data, upper=NA, lower=NA, upper.method="ml",
         lower.method="ml", plot=T, ...)
```

REQUIRED ARGUMENTS

`data:` a numeric vector or `"timeSeries"` object.

OPTIONAL ARGUMENTS

`upper:` a numeric value of upper threshold on the right tail. The default is
`NA`.

`lower:` a numeric value of lower threshold on the left tail. The default is `NA`.

`upper.method:` a character string which specifies the estimation method for the
right tail. It should be set to `"ml"` for maximum likelihood estimation
or `"lmom"` for L-Moments method. The default is `"ml"`.

`lower.method:` a character string which specifies the estimation method for the
left tail. It should be set to `"ml"` for maximum likelihood estimation
or `"lmom"` for L-Moments method. The default is `"ml"`.

`plot:` a logical value specifying if a QQplot of excesses over threshold versus
GPD quantiles with estimated shape is generated. The linear nature
of this plot supports the assumption that excesses have a GPD distri-
bution. The default is `TRUE`.

VALUE

an object of class `"gpd"` representing the fit. See `gpd.object` for
details. A QQplot of excesses over threshold versus quantiles of the
estimated GPD is generated on a graphical device if `plot=T`.

DETAILS

`gpd.tail` is named `pot.2tails.est` in the original EVANESCE li-
brary.

REFERENCES

Prescott, P., and Walden, A. T. (1980). Maximum likelihood estima-
tion of the parameters of the generalized extreme-value distribution.
Biometrika, 67(3):723-724.

Embrechts, P., Kluppelberg, C., and Mikosch, T. (1997). *Modelling Extremal Events for Insurance and Finance.* New York: Springer-Verlag.

SEE ALSO

`gpd.lmom`, `gpd.ml` ,`gpd.object` . `gpd` .

EXAMPLE

```
tmp.dat = BRAZIL.COFFEE[,2][as.matrix(seriesData(BRAZIL.COFFEE[,2])) != 0]
gpd.tail(tmp.dat, plot=F)
```

gpd Fit Generalized Pareto Distribution

DESCRIPTION

Fits a generalized Pareto (GPD) distribution to excesses over a high threshold.

```
gpd(data, threshold=NA, nextremes=NA, method="ml",
    information="observed", max.fcal=1000, max.iter=200)
```

REQUIRED ARGUMENTS

`data`: a numeric vector or `"timeSeries"` object.

`threshold`: a numeric value of threshold. Either `threshold` or `nextremes` must be given but not both. The default is `NA`.

`nextremes`: a number of upper extremes to be used. Either `nextremes` or `threshold` must be given but not both. The default is `NA`.

OPTIONAL ARGUMENTS

`method`: a character string which specifies the estimation method. It should be set to `"ml"` for maximum likelihood estimation or `"pwm"` for probability-weighted moments method. The default is `"ml"`.

`information`: a character string which specifies the method of calculating variance-covariance matrix of the parameter estimates. It can be either `"observed"` or `"expected"` information. Notice for `"pwm"` method only `"expected"` information is used if possible. The default is `"observed"`.

VALUE

an object of class `"gpd"` representing the fit. See `gpd.object` for details.

DETAILS

This function uses the non-linear minimization routine `nlmin` when `method="ml"` is chosen. It estimates one tail at a time. To estimate 2 tails simultaneously, use `gpd.tail`.

REFERENCES

Hosking, J. R. M., and Wallis, J. R. (1987). Parameter and quantile estimation for the generalized Pareto distribution. *Technometrics*, 29(3):339-349.

SEE ALSO

`plot.gpd`, `shape` , `quant` ,`gpd.object` ,`gpd.tail` .

EXAMPLE

```
# Fits GPD to excess losses over 10 for the Danish
# fire insurance data
out = gpd(danish, 10)
```

gphTest GPH Test for Long Memory or Long-Range Dependence

DESCRIPTION

Returns the GPH test for long memory or long range dependence.

```
gphTest(x, spans=1, taper=0.1, pad=0, detrend=F, demean=T,
        alpha=0.5, na.rm=F)
```

REQUIRED ARGUMENTS

x : a vector, matrix, data frame, or a `"timeSeries"` object with a numeric object in the data slot, representing either a univariate or a multivariate time series.

OPTIONAL ARGUMENTS

spans : a sequence of lengths of modified Daniell smoothers to run over the raw periodogram. Use `spans=1`, the default, for the raw periodogram. A modified Daniell smoother has all values equal except for the 2 end values which are half the size of the others. The values should be odd integers.

taper : fraction of each end of the time series that is to be tapered. A split cosine taper is applied to `taper * length(x)` points at each end of series. This must take values between 0 and 0.5. The default is 0.1.

pad : fraction of the length of x that is to be padded: `pad * length(x)` zeros are added to the end of the series before computing the periodogram. The default is 0.

detrend : a logical flag: if `TRUE`, a least squares line is removed from each component of the series before computing periodogram. The default is `FALSE`.

demean : a logical flag: if `TRUE`, the mean of each series is removed before computing the periodogram (`detrend` also removes the mean). The default is `TRUE`.

alpha : a number which is usually between 0 and 1. If x has n observations, then `n^alpha` frequencies will be used in the underlying regression. The default is 0.5.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computing the test statistics. The default is `FALSE`.

VALUE

an object of class `"gphTest"`, which contains the following components:

d : a numeric vector giving the estimates of fractional integration parameters.

std.err : a numeric vector giving the asymptotic standard errors of d.

n : an integer giving the sample size of x.

na : an integer giving the number of missing values in x.

n.freq : an integer giving the number of frequencies used in the regression. This is equal to n^alpha.

DETAILS

The function first calls spec.pgram to obtain a periodogram estimate, then log periodogram regression is used to estimate the fractional integration parameter, as proposed by Geweke and Porter-Hudak (1983).

If the input x is multivariate, then the fractional integration parameter is estimated for each series separately.

REFERENCES

Geweke, J., and Porter-Hudak, S. (1983). The estimation and application of long memory time series models. *Journal of Time Series Analysis*, 4:221-238.

SEE ALSO

FARIMA,d.ros ,d.pgram ,d.whittle ,rosTest ,spec.pgram .

EXAMPLE

```
# GPH test on absolute DELL stock returns.
gphTest(abs(dell.s))
```

gumbel.bivd.object Child Classes of bivd Class

DESCRIPTION

These are S version 4 objects of classes that inherit from `"bivd"` class, such as Gumbel, Galambos, Husler Reiss, Tawn, Frank, Kimeldorf Sampson, Joe, BB1, BB2, BB3, BB4, BB5, BB6, BB7, Normal, and Mixed Mixed Normal copula classes.

GENERATION

These classes of objects are constructed from the their respective functions. For example, for `"gumbel.bivd"` class, the constructing function is `gumbel.bivd(copula, Xmarg, Ymarg, param.Xmarg, param.Ymargin)`.

STRUCTURE

The following slots must be present in a legitimate `"bivd.copula"` class object:

`copula` : the copula class name.

`Xmarg` : abbreviated name of the marginal distribution for `X`. If the distribution name is `dist`, it is assumed that the functions `ddist`, `pdist`, and `qdist` are implemented in Splus. For example, if `Xmarg="norm"`, functions `dnorm`, `pnorm`, and `qnorm` should be available for correct performance of the bivariate distribution functions. The distribution names are not limited to those standard for Splus, assuming that the user creates appropriate density and quantile functions.

`Ymarg` : abbreviated name of the marginal distribution for `Y`.

`param.Xmargin` : Parameters for the marginal distribution of `X`. The number, and the order of the parameters should correspond to those of the functions `ddist`, `pdist`, and `qdist`.

`param.Ymargin` : parameters for the marginal distribution of `Y`.

METHODS

The `"gumbel.bivd"` class of objects currently has no methods for generic functions.

Following functions are implemented: `pbivd`, `dbivd`, `rbivd`, `persp.dbivd`, `persp.pbivd`, `contour.dbivd`, `contour.pbivd`.

SEE ALSO

`pbivd`,`contour.pbivd.object` .

gumbel.copula.object Gumbel Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"gumbel.copula"`, which inherits from `copula` and `ev.copula` classes.

GENERATION

This class of objects is constructed from the `gumbel.copula(delta)` function or `ev.copula` function.

METHODS

The `"gumbel.copula"` class of objects currently has methods for the following generic functions:

`Afunc`, `AfirstDer`, `AsecondDerPhiDer`, `Hderiv`, `pcopula`, `dcopula`, `rcopula`, `contour.plot`, `tail.index`, `Kendalls.tau`, `Spearmans.rho`, `dcdx`.

Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula`, `contour.dcopula`, `contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"gumbel.copula"` class object:

`parameters` : a numerical value for the parameter `delta` (greater or equal to 1).

`param.names` : `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (Gumbel copula family), and the copula class from which it inherits (Extreme Value Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

Gumbel, E. J. (1960). Distributions des valeurs extremes en plusiers dimensios. *Publ. Inst. Statist.*, Univ. Paris, 9:171-173.

SEE ALSO

`copula.object`,`ev.copula.object`,`ev.copula`.

gumbel Fit Gumbel Distribution

DESCRIPTION

Fits a Gumbel distribution (GEV with `xi` parameter equal to 0) to block maxima data.

```
gev(data, block=NA, max.fcal=500, max.iter=200)
```

REQUIRED ARGUMENTS

`data`: a numeric vector or `"timeSeries"` object. The interpretation of `data` depends on the value of `block`: if no block size is specified then `data` are interpreted as block maxima; if block size is set, then `data` are interpreted as raw data and block maxima are calculated internally.

OPTIONAL ARGUMENTS

`block`: an integer specifying the block size, or a character string that takes the value of `"month"`, `"quarter"`, `"semester"` or `"year"` (if `data` is a `"timeSeries"` object with calendar times), or simply `NA` when block maxima data are supplied. The default is `NA`.

`max.fcal` : an integer specifying the maximum number of function evaluations allowed in maximum likelihood estimation. The default is 500.

`max.iter` : an integer specifying the maximum number of iterations allowed in maximum likelihood estimation. The default is 200.

VALUE

an object of class `"gev"` representing the fit. See `gev.object` for details.

DETAILS

This function is primarily intended for comparison with GEV for assessing the need for a heavy-tailed Frechet (or short-tailed Weibull) to model block maxima.

SEE ALSO

`gev`, `plot.gev` ,`gev.object` .

EXAMPLE

```
# Fit Gumbel to monthly maxima of daily returns on BMW share price
out = gumbel(bmw, "month")

# Fit Gumbel to maxima of blocks of 100 observations
out = gumbel(bmw, 100)

# Fit Gumbel to the data in nidd.annual, the annual
# maximum water levels of the River Nidd
out = gumbel(nidd.annual)
```

heteroTest.OLS Use `heteroTest()` on an `OLS` Object

DESCRIPTION

This is a method for the function `heteroTest()` for objects inheriting from class `"OLS"`. See `heteroTest` for the general behavior of this function.

```
heteroTest.OLS(object, method="white", cross.terms=F,
               regressors=NULL, robust=F)
```

REQUIRED ARGUMENTS

`object` : an object of class `"OLS"`. This is usually returned by a call to `OLS` function. regression.

OPTIONAL ARGUMENTS

`method` : a character string specifying the test to compute. Valid choices are: `"white"` for White's general test for heteroskedasticity, and `"lm"` for Breusch-Pagan LM test. The default is `"white"`.

`cross.terms` : a logical flag: if `TRUE`, cross terms of the regressors will be included in the regression to compute White's test. This is ignored if `method="lm"`. The default is `FALSE`.

`regressors` : a matrix of variables with each row representing one observation. This is required if `method="lm"`, but ignored if `method="white"`.

`robust` : a logical flag: if `TRUE`, then a robust version of the Breusch-Pagan LM test will be returned. This is ignored if `method="white"`. The default is `FALSE`.

VALUE

an object of class `"heteroTest"`. See the help file for `heteroTest`.

REFERENCES

Breusch, T., and Pagan, A. (1979). A simple test for heteroskedasticity and random coefficient variation. *Econometrica*, 47:1287-1294.

White, H. (1980), A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48:817-838.

SEE ALSO

`autocorTest`,`collinearTest` , `heteroTest` .

heteroTest Test for Heteroskedasticity in Regression Residuals

DESCRIPTION

Returns the selected test statistic for heteroskedasticity given a fitted regression object.

This function is an S Version 3 generic (see `Methods`); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: `OLS`, `lm`.

```
heteroTest(x, ...)
```

REQUIRED ARGUMENTS

`x` : an object which is usually returned by a model fitting function.

VALUE

an object of class `"heteroTest"`, which inherits from `"OLS"`. The following components must be present in a legitimate `"heteroTest"` object, in addition to those present in an `"OLS"` object:

`method` : a character string describing the test computed. For example, see the help file for `heteroTest.OLS`.

`statistic` : the value of the test statistic.

`p.value` : the corresponding p-value of the test.

SEE ALSO

`archTest`,`autocorTest` ,`collinearTest` ,`heteroTest.OLS` .

highFreq3M.df Intray-day 3M Stock Prices

SUMMARY

This is a data frame representing intra-day transaction level data of 3M stock, with the following three columns:

`trade.day:` integer representing the trading day of the month.

`trade.time:` integer representing the trading time recorded as the number of seconds from midnight.

`trade.price:` transaction price in dollars.

hill Calculate and Plot Hill Estimate

DESCRIPTION

Calculates the Hill estimate of the tail index of heavy-tailed data, or of an associated quantile estimate and generates a plot optionally.

```
hill(data, option="alpha", start=15, end=NA, p=NA, ci=0.95,
    plot=T, reverse=F, auto.scale=T, labels=T, ...)
```

REQUIRED ARGUMENTS

data : a numeric vector or a `"timeSeries"` object with a numeric vector in the data slot.

OPTIONAL ARGUMENTS

option : a character string specifying whether `"alpha"`, `"xi"` (1/alpha) or `"quantile"` (a quantile estimate) should be calculated and plotted. The default is `"alpha"`.

start : an integer specifying the lowest number of order statistics to start with. The default is `15`.

end : an integer specifying the highest number of order statistics to end with, or `NA`, in which case all positive data points are used. The default is `NA`.

p : a numerical value specifying the probability required when option `"quantile"` is chosen. It is not ignored with other option values. The default is `NA`.

ci : a number between `0` and `1` specifying the probability for asymptotic confidence band. Set it to `FALSE` if no confidence band is needed. The default is `0.95`.

plot : a logical flag: if `TRUE`, a plot is generated; if `FALSE`, only a data frame of results is returned. The default is `TRUE`.

reverse : a logical flag: if `TRUE` the plot is in terms of increasing thresholds; if `FALSE`, the plot is in terms of increasing number of order statistics. The default is `FALSE`.

auto.scale : a logical flag: if `TRUE`, the plot is automatically scaled; if `FALSE`, `xlim` and `ylim` graphical parameters may be passed. The default is `TRUE`.

labels : a logical flag: if `TRUE`, axes are labeled. The default is `TRUE`.

... : any optional arguments that can be passed down to the `plot` function.

VALUE
>   a data frame which contains the following components:

`alpha, xi or quantile` : alpha, xi or quantile depending on the option supplied.

`orderStat` : the corresponding order statistics.

`threshold` : the corresponding threshold values.

DETAILS
>   This plot is usually calculated from the alpha perspective. For a generalized Pareto analysis of heavy-tailed data using the `gpd` function, it helps to plot the Hill estimates for `xi` . A plot of tail indices or quantile values over increasing thresholds or numbers of order statistics will be drawn on a graphical device if `plot=T`.

SEE ALSO
>   `shape`, `quant` .

EXAMPLE

```
# Hill plot of heavy-tailed Danish fire insurance data
hill(danish)

# Hill plot of estimated 0.999 quantile of Danish fire insurance data
hill(danish, option="quantile", end=500, p=0.999)
```

histPlot Trellis Histogram Plot

DESCRIPTION

Generates a trellis object representing the histogram of a given numeric object.

```
histPlot(x, strip.text="", n.density=50, ...)
```

REQUIRED ARGUMENTS

x : a numeric vector or a `"timeSeries"` object with a numeric vector in the data slot.

OPTIONAL ARGUMENTS

strip.text : a character string which will appear in the strip of the trellis plot. The default is "", which draws nothing in the strip.

n.density : an integer specifying the number of classes (i.e., bars) the histogram should have. The default is 50.

... : any other optional arguments that can be passed down to low level plot functions.

VALUE

an object of class `trellis`, which is automatically plotted by `print.trellis`. If this function is invoked directly without saving the returned object, a trellis device will be open to draw the plot if one is not available yet.

SEE ALSO

`histogram`,`trellis.args` .

EXAMPLE

```
histPlot(dell.s, strip="DELL Returns")
```

hpfilter   Hodrick-Prescott Filter

DESCRIPTION

Returns the Hodrick-Prescott decomposition of a macroeconomic time series into a smooth trend component and a cyclical component.

```
hpfilter(x, lambda)
```

REQUIRED ARGUMENTS

x : a vector, or a `"timeSeries"` object of the original data. Usually this is represented on a logarithmic scale.

lambda : a positive number representing the smoothness parameter. A larger number results in more smoothing. A rule of thumb is to use 100 for annual, 1600 for quarterly, and 14400 for monthly data.

VALUE

a vector, or a `"timeSeries"` with the same length as x, representing the smooth trend component. The cyclical (business cycle) component is the difference between the original data and the trend component.

REFERENCES

Hodrick, R. J., and Prescott, E. C. (1997). Postwar U.S. business cycles: an empirical investigation. *Journal of Money, Credit and Banking*, 29:1-16.

Kydland, F. E., and Prescott, E. C. (1990). Business cycles: real facts and a monetary myth. *Federal Reserve Bank of Minneapolis Quarterly Review*, Spring(1990):3-18.

SEE ALSO

`stl`.

EXAMPLE

```
# compute the cyclical component in the US CPI data.
uscpi.cyc = log(uscpi.dat) - hpfilter(log(uscpi.dat), 14400)
```

husler.reiss.copula.object Husler Reiss Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"rusler.reiss.copula"` , which inherits from `copula` and `ev.copula` classes.

GENERATION

This class of objects is constructed from the `husler.reiss.copula(delta)` function or `ev.copula` function.

METHODS

The `"husler.reiss.copula"` class of objects currently has methods for the following generic functions:

`Afunc`, `AfirstDer`, `AsecondDerPhiDer`, `Hderiv`, `pcopula`, `dcopula`, `rcopula`, `contour.plot`, `tail.index`, `Kendalls.tau`, `Spearmans.rho`, `dcdx`.

Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula`, `contour.dcopula`, `contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"husler.reiss.copula"` class object:

`parameters` : a numerical value for the parameter `delta` (greater or equal to 0).

`param.names` : `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (Husler Reiss copula family), and the copula class from which it inherits (Extreme Value Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

Husler, J., and Reiss, R.-D. (1989). Maxima of normal random vectors: between independence and complete dependence. *Statist. Probab. Lett.*, 7:283-286.

SEE ALSO

`copula.object`,`ev.copula.object` ,`ev.copula` .

IC.FARIMA   Use `IC()` on a `FARIMA` Object

```
IC.FARIMA(object, ...)
```

This is a method for the function `IC()` for objects inheriting from class `"FARIMA"`. See `IC` for the general behavior of this function and for the interpretation of `object`.

IC.OLS Use `IC()` on an `OLS` Object

      `IC.OLS(object, ...)`

This is a method for the function `IC()` for objects inheriting from class `"OLS"`. See `IC` for the general behavior of this function and for the interpretation of `object`.

IC.SEMIFAR Use `IC()` on a SEMIFAR Object

    `IC.SEMIFAR(object, ...)`

This is a method for the function `IC()` for objects inheriting from class `"SEMIFAR"`. See `IC` for the general behavior of this function and for the interpretation of `object`.

IC Model Information Criterion

DESCRIPTION

Returns the selected information criterion for a given model.

This function is an S Version 3 generic (see `Methods`); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: `OLS`, `FARIMA`, `SEMIFAR`, and `VAR`.

```
IC(object, type="BIC")
```

REQUIRED ARGUMENTS

`object` : any model object with the following components: `"loglike"` giving the log-likelihood value, `"p"` giving the number of model parameters, and `"n"` giving the sample size.

OPTIONAL ARGUMENTS

`type` : a character string giving the desired information criterion. Valid choices are: `"loglike"` for log-likelihood value, `"AIC"` for Akaike Information Criterion, `"BIC` for Bayesian Information Criterion, and `"HQ"` for Hannan-Quinn Criterion.

VALUE

a number giving the chosen information criterion.

IC.VAR Use `IC()` on a `VAR` Object

```
IC.VAR(object, ...)
```

This is a method for the function `IC()` for objects inheriting from class
`"VAR"`. See `IC` for the general behavior of this function and for the
interpretation of `object`.

iEMA.kernel Kernel Function of Moving Average Operators

DESCRIPTION

> Returns the kernel function of moving average operators for inhomogeneous or tick-by-tick time series.

> ```
> iEMA.kernel(tau, iter, max.t=10, n.points=101, plot=F, ...)
> iMA.kernel(tau, iter, max.t=10, n.points=101, plot=F, ...)
> ```

REQUIRED ARGUMENTS

  `tau` : the range for basic EMA operator when `iter=1`.

  `iter` : an integer specifying the number of times the EMA or MA operator will be iterated.

OPTIONAL ARGUMENTS

 `max.t` : an integer giving the maximum number of lags to use in the calculation. The default is 10.

 `n.points` : an integer giving the number of points to use in the calculation. In general, `n.point` equally spaced points between 0 and `max.t` are used. The default is 101.

 `plot` : a logical flag: if `TRUE`, the kernel function of the chosen operator will be plotted. The default is `FALSE`.

 `...` : any optional argument that may be passed down to the plot function.

VALUE

> a list with following components, which can be directly plotted by applying the `plot` function:

> `x`: a vector with length `n.points`. These values are equally spaced between 0 and `max.t`.

> `y`: a vector with length `n.points`. These are the values of the kernel function corresponding to `x`.

REFERENCES

> Zumbach, G., and Muller, U. A. (2001). Operators on inhomogeneous time series. *International Journal of Theoretical and Applied Finance*, 4(1):147-178.

SEE ALSO

> iEMA,iMVar .

EXAMPLE

```
# an EMA kernel with range tau*iter = 4
iEMA.kernel(1, 4, plot=T)

# a rectangular MA kernel
iMA.kernel(1, 200, plot=T)
```

iEMA Exponential Moving Average or Difference for Inhomogeneous Time
Series

DESCRIPTION

Returns the exponential moving average or difference for inhomoge-
neous or tick-by-tick time series.

```
iEMA(x, tau, time=NULL, iter=1, interp="linear", na.rm=F,
    openTime="9:30", closeTime="16:00")
iDiff(x, tau, time=NULL, iter=1, interp="linear", na.rm=F,
    openTime="9:30", closeTime="16:00", annual=F)
iMA(x, tau, time=NULL, iter=1, interp="linear", na.rm=F,
    openTime="9:30", closeTime="16:00")
```

REQUIRED ARGUMENTS

x : a numeric vector, or a `"timeSeries"` object with a numeric vector in
the data slot.

tau : the range for underlying EMA operator when `iter=1`. In general, the
range of the EMA is given by `tau*iter`.

OPTIONAL ARGUMENTS

time : the time index corresponding to `x`. If `time=NULL`, `time` will be set to
`1:length(x)` when `x` is a numeric vector, or the result of a call to
`bizTime` when `x` is a `"timeSeries"` object. A custom time index can
be passed in as a numeric vector with the same length as `x`.

iter : an integer specifying the number of times the EMA or MA operator
will be iterated. The default is 1.

interp : a character string specifying the discretization assumption when com-
puting the EMA from discrete data. Valid choices are: `"previous"`
for using the previous value, `"linear"` for linear interpolation, and
`"next"` for using the next value available. The default is `"lienar"`.

na.rm : a logical flag: if `TRUE`, missing values in `x` will be removed before
computing the EMA. The default is `FALSE`.

openTime : a character string with the format ”MM:SS” specifying the market
opening time. It is used by `bizTime` to compute the implied business
time index. If `x` is not a `"timeSeries"` object or `time` is given explic-
itly, `openTime` is ignored. The default is `"09:30"`.

closeTime : a character string with the format ”MM:SS” specifying the mar-
ket closing time. It is used by `bizTime` to compute the implied busi-
ness time index. If `x` is not a `"timeSeries"` object or `time` is given
explicitly, `closeTime` is ignored. The default is `"16:00"`.

`annual` : a logical flag: if `TRUE`, the returned object will be annualized. This is usually applied to return annualized financial returns. The default is `FALSE`.

VALUE

a numeric vector, or a `"timeSeries"` object with a numeric vector in the data slot. `iEMA` returns the EMA of `x` with the chosen range and number of iterations, `iMA` returns the MA of `x` with the chosen range and number of iterations, while `iDiff` returns the difference of inhomogeneous or tick-by-tick time series, which can be used to characterize tick-by-tick financial returns.

REFERENCES

Zumbach, G., and Muller, U. A. (2001). Operators on inhomogeneous time series. *International Journal of Theoretical and Applied Finance*, 4(1):147-178.

SEE ALSO

`EWMA`,`SMA` ,`getReturns` ,`iMVar` ,`iEMA.kernel` ,`iMA.kernel` .

EXAMPLE

```
# an EMA with range tau*iter = 4
iEMA(1:100, 2, iter=2)

# an MA with range 2*tau = 2
iMA(1:100, 1, iter=200)
```

imm.dates International Monetary Market (IMM) Dates

DESCRIPTION

Returns a `"timeDate"` vector representing the International Monetary Market (IMM) dates, which are the third Wednesday in March, June, September, and December.

```
imm.date(start, end, holidays="NYSE", following="none", ...)
```

REQUIRED ARGUMENTS

start: a `"timeDate"` object, or a string which can be parsed into a `"timeDate"` object.

end: a `"timeDate"` object, or a string which can be parsed into a `"timeDate"` object.

OPTIONAL ARGUMENTS

holidays: a character string giving the name of the holidays. This is set to `"NYSE"` by default and calls the function `holiday.NYSE` to generate the holidays. To set this to any other string, there must be a corresponding (user-supplied) `holiday.xxx` function to generate the holidays.

following: a character string specifying the action if an IMM date is a holiday. The default is `"none"`, which does not do anything special. Other valid choices are: `"previous"` which uses the previous business day; `"next"` which uses the next business day; `"modified"` which uses the next business day if it is in the same month, or the previous business day otherwise.

...: any optional arguments that may be passed to parse `start` and `end` as `"timeDate"` objects.

VALUE

a `"timeDate"` vector representing the IMM dates between `start` and `end`.

SEE ALSO

`holidays`.

import.dat Monthly Imports and Import Taxes of Argentina

SUMMARY
  The `import.dat` data frame has 96 rows and 2 columns. The sample
  runs from January 1983 to December 1990.

DATA DESCRIPTION
  This data frame contains the following columns:

taxes: monthly import taxes of Argentina.

import: monthly imports of Argentina.

impRes   Impulse Response Function

DESCRIPTION

Returns the impulse response function given a fitted VAR object.


```
impRes(x, period=NULL, std.err="none", plot=F, unbiased=T,
        order=NULL, ...)
```

REQUIRED ARGUMENTS

`x` : an object inheriting from class `"VAR"`.

OPTIONAL ARGUMENTS

`period` : the number of periods to compute the impulse respones. By default, we set `period=p*(k-1) + 1`, where `p` is the order of the autoregressive model, and `k` is the dimension of the multivariate response.

`std.err` : a character string specifying the type of standard errors to be returned. Currently the only valid choices are `"none"` and `"asymptotic"`: if `std.err="none"`, standard errors will not be computed; if `std.err="asymptotic"`, the asymptotic standard errors will be returned. The default is `"none"`.

`plot` : a logical flag: if `TRUE`, the forecast error variance decomposition will be plotted. The default is `FALSE`.

`unbiased` : a logical flag; if `TRUE`, unbiased estimate of error covariance will be used; if `FALSE`, the maximum likelihood estimate of error covariance will be used. The default is `TRUE`.

`order` : a numeric index vector, or a character vector with elements corresponding to the variable names used to fit `x`. This can be used to rearrange the order of the variables when computing forecast error variance decomposition.

`...` : any optional arguments that can be passed to `plot.impDecomp`.

VALUE

an S Version 3 object of class `"impDecomp"`, containing the following components:

`values` : an array with dimension `c(k, k, period)` representing the impulse responses, where `k` is the dimension of the multivariate response.

`std.err` : an array with dimension `c(k, k, period)` representing the standard errors of impulse response function, where `k` is the dimension of the multivariate response. This is only present if `std.err="asymp"` in the call. A plot of the impulse response function will be shown in a graphical device if `plot=T`.

REFERENCES

Lutkepohl, H. (1990). *Introduction to Multiple Time Series Analysis*, Springer-Verlag.

SEE ALSO

`fevDec`, `plot.impDecomp` , `VAR.ar2ma` .

EXAMPLE

```
pol.mod = VAR(cbind(M2,GDP,U)~ar(13), data=policy.dat)
pol.fev = fevDec(pol.mod, period=5, plot=T)
```

iMVar Moving Sample Statistics for Inhomogeneous Time Series

DESCRIPTION

Returns moving sample statistics for inhomogeneous or tick-by-tick time series.

```
iMVar(x, tau, p=2, time=NULL, iter=1, interp="linear", na.rm=F,
    openTime="9:30", closeTime="16:00")
iMSD(x, tau, p=2, time=NULL, iter=1, interp="linear", na.rm=F,
    openTime="9:30", closeTime="16:00")
iMSkewness(x, tau, p=2, time=NULL, iter=1, interp="linear", na.rm=F,
        openTime="9:30", closeTime="16:00")
iMKurtosis(x, tau, p=2, time=NULL, iter=1, interp="linear", na.rm=F,
        openTime="9:30", closeTime="16:00")
iMNorm(x, tau, p=2, time=NULL, iter=1, interp="linear", na.rm=F,
    openTime="9:30", closeTime="16:00")
iMCor(x, y, tau, p=2, time=NULL, iter=1, interp="linear", na.rm=F,
    openTime="9:30", closeTime="16:00")
```

REQUIRED ARGUMENTS

`x, y` : a numeric vector, or a `"timeSeries"` object with a numeric vector in the data slot. If `y` is also given as in `iMCor`, it must have the same length as `x`.

`tau` : the range for basic EMA operator with `iter=1`.

OPTIONAL ARGUMENTS

`p` : an integer specifying the power to use when computing the moving sample variance. The default is 2.

`time` : the time index corresponding to `x`. If `time=NULL`, `time` will be set to `1:length(x)` when `x` is a numeric vector, or the result of a call to `bizTime` when `x` is a `"timeSeries"` object. A custom time index can be passed in as a numeric vector with the same length as `x`.

`iter` : an integer specifying the number of times the MA operator will be iterated. The default is 1.

`interp` : a character string specifying the discretization assumption when computing the EMA from discrete data. Valid choices are: `"previous"` for using the previous value, `"linear"` for linear interpolation, and `"next"` for using the next value available. The default is `"lienar"`.

`na.rm` : a logical flag: if `TRUE`, missing values in `x` will be removed before computing the EMA. The default is `FALSE`.

`openTime` : a character string with the format "MM:SS" specifying the market opening time. It is used by `bizTime` to compute the implied business time index. If `x` is not a `"timeSeries"` object or `time` is given explicitly, `openTime` is ignored. The default is `"09:30"`.

`closeTime` : a character string with the format "MM:SS" specifying the market closing time. It is used by `bizTime` to compute the implied business time index. If `x` is not a `"timeSeries"` object or `time` is given explicitly, `closeTime` is ignored. The default is `"16:00"`.

VALUE

a numeric vector, or a `"timeSeries"` object with a numeric vector in the data slot. `iMVar` returns the moving variance of `x` with order `p`, `iMSD` returns the moving standard deviation, `iMSkewness` returns the moving skewness, `iMKurtosis` returns the moving kurtosis, `iMNorm` returns the moving norm, and `iMCor` returns the moving sample correlation.

DETAILS

Except for `iMNorm`, all these functions remove the first moving sample statistic because the first moving variance is zero due to the fact that we use the first observation to start the basic EMA.

REFERENCES

Zumbach, G., and Muller, U. A. (2001). Operators on inhomogeneous time series. *International Journal of Theoretical and Applied Finance*, 4(1):147-178.

SEE ALSO

`EWMA`,`SMA` ,`iEMA` ,`rollVar` .

EXAMPLE

```
# Moving variance with tau=1
iMVar(1:100, 1)

# Moving variance with tau=2
iMVar(1:100, 2)
```

interpNA   Missing Values Interpolation

DESCRIPTION

Interpolates missing values in a data set.

        interpNA(x, method="spline", maxStartNA=5)

REQUIRED ARGUMENTS

x: a vector, matrix, data frame, or a "timeSeries" object with some
missing values (NA).

OPTIONAL ARGUMENTS

method: a character string specifying the method used for interpolation. Valid
choices are: "before", "after", "nearest", "linear", and "spline".
See Details section for the explanations. The default is "spline".

maxStartNA: an integer specifying the maximum number of missing values which
is allowed at the beginning of the time series. If there are more than
maxStartNA missing values at the beginning, an error message will be
generated; otherwise, the missing values are left as is. The default is
set to 5.

VALUE

a vector, matrix, or a "timeSeries" object which is the same as x,
except that the missing values (NA) are replaced with interpolations.
If there are fewer than maxStartNA missing values at the beginning of
the time series, those missing values are left as is.

DETAILS

If method="spline", interpNA calls the function cspline for cubic
spline interpolation. Otherwise, it calls align to use other methods.

SEE ALSO

align, cspline .

EXAMPLE

        djia.close = djia[positions(djia) >= timeDate("01/01/1990"), "close"]
        djia.close = interpNA(djia.close)

interpret.gpdbiv Interpretation of a Bivariate GPD Fit

DESCRIPTION

Interprets the results of a fitted bivariate GPD model using the bivariate POT method.

```
interpret.gpdbiv(out, x, y)
```

REQUIRED ARGUMENTS

out: a "gpdbiv" object as returned by the gpdbiv function.

x: a numeric value (greater than first threshold) specifying the extreme level of interest for dataset 1.

y: a numeric value (greater than second threshold) specifying the extreme level of interest for dataset 2.

VALUE

a numeric vector of marginal probability of x in dataset1, marginal probability of y in dataset2, joint probability of x and y, conditional probability of y (on x), and conditional probability of x (on y). A simple report of interpretation of the fit in terms of exceedance probabilities for the point (x,y) is printed.

SEE ALSO

gpdbiv, plot.gpdbiv .

EXAMPLE

```
out = gpdbiv(-bmw, -siemens, ne1=100, ne2=100)
# probabilities of 5% falls in BMW and Siemens stock prices
interpret.gpdbiv(out, 0.05, 0.05)
```

IP.dat Seasonally Adjusted U.S. Industrial Production Index

SUMMARY

This is a monthly `"timeSeries"` object from January 1919 to November 2001, representing seasonally adjusted U.S. Industrial Production Index.

is.weekend Weekday, Weekend or Business Day Information

DESCRIPTION

Returns a logical vector denoting whether or not a `"timeDate"` object is a weekday, weekend or business day.

```
is.weekend(x)
is.weekday(x)
is.bizday(x, holidays="NYSE")
```

REQUIRED ARGUMENTS

`x`: a `"timeDate"` object, or a vector of strings which can be parsed into a `"timeDate"` object.

OPTIONAL ARGUMENTS

`holidays`: a character string giving the name of the holidays. This is set to `"NYSE"` by default and calls the function `holiday.NYSE` to generate the holidays. To set this to any other string, there must be a corresponding (user-supplied) `holiday.xxx` function to generate the holidays.

VALUE

a logical vector denoting which element of `x` is a weekday, falls on a weekend, or is a business day.

SEE ALSO

`holidays`,`weekdays` .

joe.copula.object   Joe Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"joe.copula"` , which inherits from `copula` and `archm.copula` classes.

GENERATION

This class of objects is constructed from the `joe.copula(theta)` function or `archm.copula` function.

METHODS

The `"joe.copula"` class of objects currently has methods for the following generic functions:

`PHI`, `PhiDer`, `InvPhi`, `InvPhisecondDer`, `pcopula`, `dcopula`, `rcopula`, `contour.plot`, `LAMBDA<code>`, `<code>Kendalls.tau`, `Spearmans.rho`, `dcdx`.

Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula`, `contour.dcopula`, `contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"joe.copula"` class object:

`parameters` : value for the parameter `theta` (greater or equal than 1).

`param.names` : `theta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (Joe copula family), and the copula class from which it inherits (Archimedean Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object`,`archm.copula.object` ,`archm.copula` .

KalmanFil Kalman Filtering

DESCRIPTION

Performs Kalman filtering given the response variables or the observables and a state space form representation.


```
KalmanFil(mY, ssf, ikf=NULL, task="KFLIK")
```

REQUIRED ARGUMENTS

mY : a rectangular numeric object which represents the response variables or the observables. Note that for multivariate response, the variables must be in columns.

ssf : a list which either contains the minimal necessary components for a state space form or is a valid `"ssf"` object.

OPTIONAL ARGUMENTS

ikf : an object returned by a call to `KalmanIni` function. If `NULL`, it will be constructed by `KalmanFil` automatically. If the same call to `KalmanFil` has to be made repeatedly, it is more efficient to construct and pass `ikf` before calling `KalmanFil` function.

task : a character string which specifies the purpose of the call to `KalmanFil` function. The default is `"KFLIK"`, which performs the traditional Kalman filtering and calculates the associated log-likelihood value using prediction error decomposition. Other valid choices are: `"STPRED"` for state prediction, `"STSMO"` for state smoothing, `"STFIL"` for state filtering, `"STSIM"` for state simulation, `"DSSMO"` for disturbance smoothing, and `"DSSIM"` for disturbance simulation.

VALUE

an S version 3 object of class `"KalmanFil"` which contains the following components:

call: an image of the original function call.

innov: a numeric vector representing the measurement equation innovation for a univariate model, or a numeric matrix representing the multivariate innovations for a multivariate model.

std.innov: the standardized innovations. They are equal to `innov` divided by the corresponding standard errors.

loglike: the log-likelihood value of the state space model with data `mY`.

loglike.conc: the concentrated log-likelihood value of the state space model with data `mY`.

dVar: the variance of the prediction errors. This should be close to one if maximum likelihood estimates (MLE) of the model parameters are used.

task: a character string which is the same as the `task` argument.

err: a logical flag: if `TRUE`, the Kalman filtering was successful; if `FALSE`, the Kalman filtering failed.

mOut: a `(cT*cY)` x `(cSt+2)` matrix which contains all the necessary output of Kalman filtering, with `cT` being the number of sample observations, `cY` the number of response variables or observables, and `cSt` the number of state variables. The first column corresponds to the measurement equation innovations, the last column corresponds to the inverse of the variance of the innovations, and the rest correspond to the gain matrix.

mGain: a `(cT*cY)` x `(cSt+1)` matrix which is the same as `mOut` without its first column.

mEst: a `cT` x `(cSt+cY)` matrix which contains the required smoothed variables, with `cT` being the number of sample observations, `cSt` the number of state variables, and `cY` the number of response variables or observables. This is only returned if `task` is `"STSMO"`, `"STPRED"`, `"STFIL"`, or `"STSIM"`.

positions: a `"timeDate"` vector giving the positions of the input data `mY`. This is only returned if `mY` is a `"timeSeries"` object.

DETAILS

Currently only `print` and `plot` methods are available for an object of class `"KalmanFil"` .

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

SEE ALSO

CheckSsf,KalmanIni ,KalmanSmo .

EXAMPLE

```
ssf.mod = GetSsfStsm(irregular=sqrt(15099), level=sqrt(1469))
KalmanFil(nile.dat, ssf.mod)

ssf.mod = GetSsfStsm(irregular=sqrt(15099), level=sqrt(1469))
stsm.ikf = KalmanIni(nile.dat, ssf.mod, task="STSIM")
KalmanFil(nile.dat, ssf.mod, ikf=stsm.ikf, task="STSIM")
```

KalmanIni Initialize Kalman Filtering

DESCRIPTION

Returns a list which represents the sparseness of the state space model and its initial conditions.

```
KalmanIni(mY, ssf, task="KFLIK")
```

REQUIRED ARGUMENTS

mY : a rectangular numeric object which represents the response variables or the observables. Note that for multivariate response, the variables must be in columns.

ssf : a list which either contains the minimal necessary components for a state space form or is a valid `"ssf"` object.

OPTIONAL ARGUMENTS

task : a character string which specifies the purpose of the call to `KalmanFil` function. The default is `"KFLIK"`, which performs the traditional Kalman filtering and calculates the associated log-likelihood value using prediction error decomposition. Other valid choices are: `"STPRED"` for state prediction, `"STSMO"` for state smoothing, `"STFIL"` for state filtering, `"STSIM"` for state simulation, `"DSSMO"` for disturbance smoothing, and `"DSSIM"` for disturbance simulation.

VALUE

a list which contains the following components:

cIndex, mIndex: these represent the sparseness of the coefficient matrix of the state space model.

cIni, mIni: these represent the exact initial conditions of the state space model.

mIniP, cOffP, mOffP: these represent the exact initial conditions of the covariance matrix of state variables. They are only returned when `task` is `"STSMO"` or `"STSIM"`.

DETAILS

A call to `KalmanIni` must precede a call to `KalmanFil` for state space modeling. Usually this is conducted internally so that a user does not have to call `KalmanIni` explicitly. However, when the same state space modeling function has to be called repeatedly with different parameter values, such as in maximum likelihood estimation, it is more efficient to call `KalmanIni` explicitly and then pass the returned object to the appropriate state space modeling function, such as `KalmanFil`, `SsfLoglike`, `SsfMomentEst`, and `SsfCondDens`.

REFERENCES

> Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

SEE ALSO

> CheckSsf,KalmanFil ,SsfCondDens ,SsfLoglike ,SsfMomentEst .

EXAMPLE

```
ssf.mod = GetSsfStsm(irregular=sqrt(15099), level=sqrt(1469))
stsm.ikf = KalmanIni(nile.dat, ssf.mod, task="STSIM")
KalmanFil(nile.dat, ssf.mod, ikf=stsm.ikf, task="STSIM")
```

KalmanSmo Kalman Smoothing

DESCRIPTION
>
> Performs Kalman smoothing given a Kalman filtering object and its associated state space form.

>   KalmanSmo(kf, ssf)

REQUIRED ARGUMENTS

> kf : an object of class `"KalmanFil"` as returned by a call to `KalmanFil`.

> ssf : a list which either contains the minimal necessary components for a state space form or is a valid `"ssf"` object.

VALUE
>
> an S version 3 object of class `"KalmanSmo"` which contains the following objects:

> call: an image of the original function call.

`state.residuals`: a numeric vector or matrix which represents the smoothing residuals from the state equation.

`response.residuals`: a numeric vector or matrix which represents the smoothing residuals from the response equation, or measurement equation.

`state.variance`: a numeric vector or matrix which represents the variance of `state.residuals`.

`response.variance`: a numeric vector or matrix which represents the variance of `response.residuals`.

`aux.residuals`: a numeric matrix which represents the standardized smoothing residuals.

> scores: a numeric matrix representing the analytical scores for the parameters in disturbance covariance matrix.

`positions`: a `"timeDate"` vector giving the positions of the original input data `mY`. This is only returned if `mY` is a `"timeSeries"` object.

DETAILS
>
> Currently only `print` and `plot` methods are available for an object of class `"KalmanSmo"` .

> Note that `KalmanSmo` only returns the smoothing residuals. If you need other variables such as the filtered state variables, smoothed state variables, etc, use `SsfCondDens` or `SsfMomentEst`.

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

SEE ALSO

KalmanFil,SsfCondDens ,SsfMomentEst .

EXAMPLE

```
ssf.mod = GetSsfStsm(irregular=sqrt(15099), level=sqrt(1469))
nile.fil = KalmanFil(nile.dat, ssf.mod)
KalmanSmo(nile.fil, ssf.mod)
```

Kendalls.tau   Kendalls.tau of a copula

DESCRIPTION

Computes Kendall's tau for parametric copula or an empirical copula.

```
Kendalls.tau(copula, tol=1e-5)
```

REQUIRED ARGUMENTS

`copula:` an object of class `"copula"`.

`tol:` a numerical value defining the absolute tolerance for the integral if numerical integration is required. The default is `10^-5`.

VALUE

Kendall's tau for the copula.

DETAILS

For `normal.mix` copula family the computation of Kendall's tau requires a numerical double integration, which might take a great deal of computer resources and time. It is suggested that `tol` is increased to `10^-2` for this family.

This is a generic function. All copula classes (including empirical copula) have their method functions.

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object`, `Spearmans.rho` .

EXAMPLE

```
# This example compares the Kendall's tau of an empirical copula
# from generated data, with that of the parametric copula
# that was used for data generation
ec = empirical.copula(rcopula(bb5.copula(1.5,0.7), 1000))
Kendalls.tau(ec)
Kendalls.tau(bb5.copula(1.5,0.7))
```

kimeldorf.sampson.copula.object Kimeldorf Sampson Copula Class
Object

DESCRIPTION

These are S version 4 objects of class `"kimeldorf.sampson.copula"`
, which inherits from `copula` and `archm.copula` classes.

GENERATION

This class of objects is constructed from the `kimeldorf.sampson.copula(delta)`
function or `archm.copula` function.

METHODS

The `"kimeldorf.sampson.copula"` class of objects currently has meth-
ods for the following generic functions:

PHI, PhiDer, pcopula, dcopula, rcopula, contour.plot, LAMBDA<code>,
<code>CondCinverse, Kendalls.tau, Spearmans.rho, dcdx.

Furthermore, following functions are implemented for this class: `persp.dcopula`,
`persp.pcopula, contour.dcopula, contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"kimeldorf.sampson.copula"`
class object:

`parameters` : value for the parameter `delta` (greater than 0).

`param.names` : `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, con-
taining the values of the lower bounds for the parameter(s). These
values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for
the parameter(s).

`message` : a message specifying the name of the parametric copula family (Kimel-
dorf Sampson copula family), and the copula class from which it in-
herits (Archimedean Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. Lon-
don: Chapman & Hall.

Kimeldorf, G., and Sampson, A. R. (1975). Uniform representations
of bivariate distributions. *Communications in Statistics*, 4:617-627.

SEE ALSO

copula.object,archm.copula.object ,archm.copula .

lexrates.dat Spot and Forward Exchange Rate Data in Logarithmic Scale

SUMMARY

>   This is a monthly `"timeSeries"` object from February 1976 to June 1996, with the following twelve columns:

`USCNS`: the log spot exchange rate between U.S. Dollar and Canadian Dollar.

`USCNF`: the log forward exchange rate between U.S. Dollar and Canadian Dollar.

`USDMS`: the log spot exchange rate between U.S. Dollar and Deutsche Mark.

`USDMF`: the log forward exchange rate between U.S. Dollar and Deutsche Mark.

`USFRS`: the log spot exchange rate between U.S. Dollar and French Franc.

`USFRF`: the log forward exchange rate between U.S. Dollar and French Franc.

`USILS`: the log spot exchange rate between U.S. Dollar and Italian Lira.

`USILF`: the log forward exchange rate between U.S. Dollar and Italian Lira.

`USJYS`: the log spot exchange rate between U.S. Dollar and Japanese Yen.

`USJYF`: the log forward exchange rate between U.S. Dollar and Japanese Yen.

`USUKS`: the log spot exchange rate between U.S. Dollar and British Pound.

`USUKF`: the log forward exchange rate between U.S. Dollar and British Pound.

SOURCE

>   Eric, Z. (2000). Cointegration and forward and spot exchange rate regressions. *Journal of International Money and Finance*, 19(6):785-812. 6:387-401.

list2mat Conversion Between a List and a Matrix

DESCRIPTION

Converts a list to a matrix, or a matrix to a list. These two functions are utility functions for GARCH functions; they are not intended for users to use.

```
list2mat(x, N, M, type=1, N1=N)
mat2list(x, N, L=NULL, type=1, symm=F, N1=N)
```

SEE ALSO

array2list, mat2vec.

loadings.mfactor   Use `loadings()` on an `mfactor` Object

DESCRIPTION
>   Returns selected loadings component from an `"mfactor"` object.

>   `loadings.mfactor(object, variables)`

REQUIRED ARGUMENTS

`object` : an object of class `"mfactor"`. This is usually returned by a call to `mfactor` function.

OPTIONAL ARGUMENTS

`variables` : an integer vector, or a character vector giving the indices of the loadings to be returned. If `variables` is a character vector, it must correspond to the names of the loadings.

VALUE
>   the selected loadings component of `object`.

SEE ALSO
>   `factors`,`loadings` ,`mfactor.object` ,`mfactor.r2` .

lower.mat.index Generate Lower Triangular Matrix of Indices

DESCRIPTION

Creates a lower triagular matrix of indices.

`lower.mat.index(N)`

REQUIRED ARGUMENTS

`N`: dimension.

VALUE

a matrix of lower triangular structure containing indices.

SEE ALSO

`mat2vec`.

lutkepohl.e1 Quarterly West German Macroeconomic Data

SUMMARY

This is a quarterly `"timeSeries"` object from 1960 to 1982 with the
following four columns, as appeared in Table E.1 in Luktepohl (1990):

invest: seasonally adjusted West German fixed investment in billions of Ger-
man Mark.

income: seasonally adjusted West German disposable income in billions of Ger-
man Mark.

cons: seasonally adjusted West German consumption expenditures in bil-
lions of German Mark.

SOURCE

Lutkepohl, H. (1990). *Introduction to Multiple Time Series Analysis*,
Springer-Verlag.

mat2vec Conversion Between a Matrix and a Vector

DESCRIPTION

Puts the lower triangular elements of a matrix into a vector, or vice versa.

```
mat2vec(x, type=1)
vec2mat(x, N, type, symm=F, add.diag=NULL)
```

REQUIRED ARGUMENTS

x: a matrix.

type: 1 indicates to put the lower triangular elements of x into a vector. 2 indicates to put the diagonal elements of x into a vector and fill in zeros if necessary.

VALUE

a vector specified above.

SEE ALSO

`array2list`, `list2mat`.

mean.eq Function to Interpret Conditional Mean Specification

DESCRIPTION

Translates GARCH/MGARCH mean equation.

```
mean.eq(x, N=1, armaType="diag")
```

REQUIRED ARGUMENTS

x: one of the following or a combination of them: `1`, `-1`, `arma(r,s)`, `ar(r)`, `ma(s)`, `sd.in.mean`, `var.in.mean`, `logvar.in.mean`.

OPTIONAL ARGUMENTS

N: the dimension of the series.

armaType: character vector, specifying the type of coefficient matrices for ARMA terms in the multivariate cases. Valid choices are `"diag"`, `"lower"` or `"full"`.

VALUE

a list specifying the mean equation in the model.

SEE ALSO

`garch.eq`, `mgarch.eq`.

meplot Calculate and Plot Sample Mean Excesses

DESCRIPTION

Calculate sample mean excesses over increasing thresholds and generates a plot of correspondingly.

```
meplot(data, omit=3, labels=T, plot=T, ...)
```

REQUIRED ARGUMENTS

data : a numeric vector or a `"timeSeries"` object with a numeric vector in the data slot.

OPTIONAL ARGUMENTS

omit : an integer specifying the number of upper plotting points to be omitted. The default is 3.

labels : a logical flag: if `TRUE`, both axes are labeled. This is ignored if `plot=F`. The default is `TRUE`.

plot : a logical flag: if `TRUE`, a plot of sample mean excesses is created. The default is `TRUE`.

... : any optional arguments that can be passed down to the `plot` function.

VALUE

a data frame which contains the following components:

threshold : the increasing thresholds.

me : the sample mean excesses over `threshold`.

DETAILS

An upward trend in the plot shows heavy-tailed behavior. In particular, a straight line with positive gradient above some threshold is a sign of Pareto behavior in tail. A downward trend shows thin-tailed behavior whereas a line with zero gradient shows an exponential tail.

Because upper plotting points are the averages of a handful of extreme excesses, these may be omitted for a prettier plot.

A plot of sample mean excesses over increasing thresholds will be drawn on a graphical device if `plot=T`.

REFERENCES

Embrechts, P., Kluppelberg, C., and Mikosch, T. (1997). *Modelling Extremal Events for Insurance and Finance*. Springer-Verlag.

SEE ALSO

emplot,qplot ,gpd .

EXAMPLE

```
# Sample mean excess plot of heavy-tailed Danish fire insurance data
meplot(danish)
```

mfactor.object Statistical Multi-Factor Model Objects

DESCRIPTION

These are objects of class `"mfactor"` which represent the fit of a statistical multi-factor model.

GENERATION

This class of objects is returned from the `mfactor` function.

METHODS

The `"mfactor"` class of objects currently has methods for the following generic functions:

`factors`, `loadings`, `plot`, `print`, `residuals`, `vcov`.

STRUCTURE

The following components must be present in a legitimate `"mfactor"` object:

call: an image of the call that produced the object.

factors: a matrix, or a `"timeSeries"` object with a matrix in the data slot, which represents the factor returns or factor scores with the factors in columns.

loadings: a matrix giving the factor loadings with the loadings for each variable in column.

k: an integer giving the number of factors used.

alpha: a numeric vector giving the "alpha", or the constants in the factor regressions for all the variables.

Omega: the variance-covariance matrix of the original data according to the fitted multi-factor model.

r2: a numeric vector with length 6 giving the following summary statistics of the factor regression r-squared: the minimum, the first quartile, median, mean, the third quartile and the maximum.

eigen: a numeric vector giving the eigenvalues of the underlying principal component analysis (PCA). If classical PCA is used, the length is equal to the number of variables; if asymptotic PCA is used, the length is equal to the number of observations.

sum.loadings: a matrix of dimension `k x 6` giving the following summary statistics for each factor loadings: the minimum, the first quartile, median, mean, the third quartile and the maximum.

SEE ALSO

    `factors, loadings.mfactor , mfactor ,mfactor.r2 ,vcov.mfactor`
    .

mfactor.r2 R-squared of Factor Regressions

DESCRIPTION

Returns the goodness-of-fit measure R-squared for all the factor regressions in a multi-factor model.

```
mfactor.r2(object, adjusted=F)
```

REQUIRED ARGUMENTS

object : an object of class `"mfactor"`. This is usually returned by a call to mfactor.

OPTIONAL ARGUMENTS

adjusted : a logical flag: if `TRUE`, the values for adjusted R-squared are returned. The default is `FALSE`.

VALUE

a numeric vector giving the R-squared for all the factor regressions.

SEE ALSO

`factors`,`loadings` ,`mfactor.object` ,`mimic` .

EXAMPLE

```
folio.mf = mfactor(folio.dat, 5)
summary(mfactor.r2(folio.mf))
```

mfactor Statistical Multi-Factor Model

DESCRIPTION

Fits a statistical multi-factor model using either classical principal component or asymptotic principal component analysis.

```
mfactor(x, k=1, refine=T, check=F, max.k=NULL, sig=0.05,
        na.rm=F)
```

REQUIRED ARGUMENTS

x: a matrix, data frame, or a `"timeSeries"` object which represents a multivariate data set, with variables in columns.

OPTIONAL ARGUMENTS

k: an integer specifying the number of factors to use, or a character string specifying the type of automatic factor selection procedure. If `k` is a character string, valid choices are: `"bn"` for Bai-Ng test, and `"ck"` for Connor-Korajczyk test. Note that automatic factor selection is only available when the number of variables is greater than the number of observations. The default is 1.

refine: a logical flag: if `TRUE`, asymptotic principal component analysis will be refined one more step. The default is `TRUE`.

check: a logical flag: if `TRUE`, the function will check if any variable in `x` has the same value for all the observations and return a list of `NA`s in that case. The default is `FALSE`.

max.k: an integer giving the maximum number of factors to try when `k` is a character string. The default is set to `min(10, T-1)` where `T` is the number of observations in `x`.

sig: the level of significance for Connor-Korajczyk test. The default is 0.05.

na.rm: a logical flag: if `TRUE`, missing values will be removed before fitting the model. The default is `FALSE`.

VALUE

an object of class `"mfactor"` representing the multi-factor model. See `mfactor.object` for details.

DETAILS

If the number of variables is smaller than the number of observations, classical principal component analysis is used. If the number of variables is larger than the number of observations, asymptotic principal

component analysis is employed. If asymptotic principal component analysis is used, either Bai-Ng test or Connor-Korajczyk test can be used to select the optimal number of factors.

REFERENCES

Bai, J., and Ng, S. (2002). Determining the number of factors in approximate factor models. *Econometrica*, 70(1):191-221.

Connor, G., and Korajczyk, R. (1988). Risk and return in an equilibrium APT: application of a new test methodology. *Journal of Financial Economics*, 21(2):255-289.

Connor, G., and Korajczyk, R. (1993). A test for the number of factors in an approximate factor model. *Journal of Finance*, 48(4):1263-1291.

SEE ALSO

`factors.mfactor`,`loadings`,`mfactor.object`,`mfactor.r2`,`mimic`,`princomp`
.

EXAMPLE

```
mfactor(folio.dat, 2)
```

mgarch.eq Function to Interpret Conditional Variance Specification

DESCRIPTION

Translates `mgarch` variance equation.

```
mgarch.eq(eq, leverage=F)
```

REQUIRED ARGUMENTS

`eq`: one of the terms that specifies the conditional variance equation in a multivariate GARCH model. See the help file for `mgarch` for more details.

`leverage`: logical variable. `TRUE` indicates to include a leverage term.

VALUE

a list specifying the mean equation in a multivariate GARCH model.

SEE ALSO

`fgarch.eq`, `garch.eq` , `mean.eq` .

mgarch.model Create A Multivariate GARCH Model Object

DESCRIPTION

Converts formula specifications into a model list that can be used by the functions `mgarch` and `simulate.mgarch`. Used primarily by `mgarch` and by users to create a model structure to set particular model parameter values.

```
mgarch.model(formula.mean, formula.var, leverage=F, y.dim=1,
    autoinit=T, armaType="diag")
```

REQUIRED ARGUMENTS

formula.mean: a formula specifying the conditional mean equation structure in the form of y~arma(p,q), where `y` is a matrix of time series and `arma(p,q)` can be replaced by `ar(p)`, `ma(q)`. If a constant term is included, `1+arma(p,q)` can be used. Exogenous variables are also allowed in the formula.

formula.var: a formula specifying the conditional variance equation structure, which can be one of ~dvec.x.y(p,q) or ~dvecac.x.y(p,q), with `x` and `y` being `mat`, `vec`, or `scalar`. The other alternatives are '~dvec(p,q)', '~ewma1', '~ewma2', '~bekk(p,q)', '~diag.g(p,q)', ~ccc.g(p,q), ~prcomp.g(p,q), with `g` being one of the univariate models given by `garch(p,q)`, `egarch(p,q)`, `pgarch(p,q)`, `tgarch(p,q)`, `two.comp(type,d)`.

OPTIONAL ARGUMENTS

leverage: a logical flag: if `TRUE`, leverage effects are included in the model. The default is `FALSE`.

autoinit: a logical flag: if `TRUE`, use the internal program to set initial values. These values are believed to be quite good for financial return time series, but they are probably not good initial guesses for other types of time series.

y.dim: the number of time series in a multivariate model.

armaType: character vector, specifying the type of coefficient matrices for ARMA terms. Valid choices are `"diag"` for diagonal matrices, `"lower"` for lower triangular matricies, or `"full"` for unrestricted matrices.

VALUE

an object of class `"mgarch.model"`.

The conditional mean equation is specified by

c.which: a logical variable. `TRUE` indicates that a constant vector is included in the conditional mean equation.

c.value : a numeric vector containing values for the constant vector term in the conditional mean equation as initial values for the optimizer.

AR: a list containing the following three components: `order` giving the order of the vector AR component; `which` a list of logical matrices of length equal to `order` indicating which coefficients are to be estimated; and `value` a list of numerical matrices giving the initial values of the vector AR coefficients as initial values for the optimizer.

MA : a list containing the following three components: `order` giving the order of the vector MA component; `which` a list of logical matrices of length equal to `order` indicating which coefficients are to be estimated; and `value` a list of numerical matrices as initial values of the vector MA coefficients.

X: a list containing the following components: `term`, giving the term labels for exogenous variables in the mean equation; and `value` , a numerical vector giving the initial values of the coefficients for exogenous variables.

The variance equation is specified by

a.value : a matrix giving the initial values for the constant term in the variance equation.

arch: a list containing the following five components: `order` giving the order of the garch moving average; `which` a list of logical matrices of length `order` indicating which coefficients are to be estimated; `value` a list of numeric matrices giving values of the coefficients; `lev.which` a logical vector of length `order` indicating the lag steps with a leverage effect for the univariate models involved; and `lev.value` a numeric vector of length `order` giving the initial values of the leverage effects.

garch: a list containing the following three components: `order` giving the order of the garch autoregression; `which` a list of logical matrices of length `order` indicating the coefficients to be estimated; and `value` a list of numeric matrices of length `order` giving values of the coefficients.

Z: a list containing the following components: `term`, giving the term labels for exogenous variables in the variance equation; and `value` , a numerical vector giving the initial values of the coefficients for exogenous variables.

power.which : logical, for `imod=3` (PARCH) only. This specifies the power for the univariate models involved in the multivariate models. `power.value`

is the initial value of the power if `power.which=TRUE`. When `power.which=FALSE`
, it indicates not to estimate the power value.

Internal model names not needed by the users

`idmod` : integer of 1 through 25 and 101.

`imod` integer of 1, 2, 3, 4, 5, 11, 12, 13, 14, 21, 22, 23, and 24.

`inpar` integer. 1 indicates to use the internal program to set default
initial values, otherwise use the user supplied initial values.

SEE ALSO

`garch.model`, `fgarch.model` .

EXAMPLE

`bekk11.mod = mgarch.model(~arma(1, 1), ~bekk(1, 1), y.dim=2)`

mgarch Fit Multivariate GARCH Model

DESCRIPTION

Fits one of a wide variety of multivariate GARCH models, including not only the basic diagonal vec models of Bollerslev, Engle, and Nelson (1994), but also the complete set of models given in Ding (1994).

```
mgarch(formula.mean=~ arma(0,0), formula.var=~ garch(0,0),
        series=NULL, series.start=1, x=NULL, x.start=1,
        xlag=0, z=NULL, z.start=1, model=NULL, leverage=F,
        n.predict=1, trace=T, cond.dist="gaussian",
        dist.par=NULL, dist.est=T, cccor.choice=1,
        cccor.value=NULL, armaType="diag", control=NULL,
        algorithm="bhhh", ...)
```

REQUIRED ARGUMENTS

`formula.mean`: a formula specifying the conditional mean equation structure of ARMA form. Can have the form `y~arma(p,q)`, where y is a matrix of time series observations. In the formula, `arma(p,q)` can be replaced by `ar(p)`, `ma(q)`. If a mean term is to be removed, `-1+arma(p,q)` must be used. Also in the `formula.mean`, y can be present or absent. In the case y does not appear, it must be specified by the `series` argument. Exogenous variables are also allowed.

`formula.var`: a formula specifying a GARCH model to fit, namely one of the following:

~`dvec(p,q)`, the basic diagonal vec model that does not force positive definiteness of the conditional covariance matrices.

~`dvec.x.y(p,q)`, with x and y being either `mat`, `vec`, or `scalar`. For example, the matrix diagonal model can be specified by ~`dvec.mat.mat(1,1)`.

~`dvecac.x.y(p,q)`, with x and y being either `mat`, `vec`, or `scalar`. There is a correspondence between models in the group `dvec` and those in the group `dvecac`. The difference is that a model in the group `dvecac` replaces the constant matrix `C` by its asymptotically estimated form.

~`ewma1`, exponentially weighted model of type 1.

~`ewma2`, exponentially weighted model of type 2.

~`bekk(p,q)`, the BEKK model.

~`diag.g(p,q)`, a collection of uncorrelated univariate GARCH models, with g being a name of one of the univariate models, e.g., ~`diag.egarch(2,2)`.

~`ccc.g(p,q)`, the conditional constant correlation model, with g being a name of one of the univariate models, e.g., ~`ccc.tgarch(2,2)`.

~`prcomp.g(p,q)`, the principal components model, with g being a name of one of the univariate models, e.g., ~`prcomp.pgarch(2,2)`.

In the three models above, if the univariate `garch(p,q)` is used, there is no need to specify g, e.g., `diag(p,q)` uses a `garch(p,q)` model. Exogenous variables are also allowed in the formula.

`series`: a matrix of time series observations. The number of columns of the matrix `series` is the number of time series to be modeled and the number of rows of the `series` is the length of the time series observations. If `y` is a vector, a univariate model is fitted.

`model`: a list specifying the model as returned by `mgarch.model`. The required arguments are `formula.mean` and `formula.var`, or `model`. If both `formula.mean` and `formula.var` are specified by the user, `mgarch` will call `mgarch.model` to generate a model list specification. If one but not both of `formula.mean` or `formula.var` is missing, the missing equation will be specified as the default: `arma(0,0)` for the mean equation, and `garch(0,0)` for the variance equation. If neither `formula.mean` nor `formula.var` is specified by the user, then a model list with the components specified as in the object created by `mgarch.model` must be supplied through the `model` argument, which has the default `NULL`. If the matrix of time series `y` in `formula.mean` is not specified, then it must be supplied in the argument `series`, which also has the default value `NULL`.

OPTIONAL ARGUMENTS

`series.start`: the element of `series` to begin with. For example, if `series.start=10`, the observations of `series` will start from the tenth row of `series`.

`x`: a vector or matrix of exogenous variables in the mean equation. The default is `NULL`. Preferably this should be specified in `formula.mean`.

`x.start`: the element of `x` to begin with. For example, if `x.start=10`, the observations of `x` will start from the tenth row of `x`.

`xlag`: maximum number of lags in exogenous variables `x`.

`z`: a vector or matrix of exogenous variables in the variance equation. Preferably this should be specified in `formula.var`.

z.start: the element of z to begin with. For example, if z.start=10, the observations of z will start from the tenth row of z.

leverage: logical flag: if TRUE, a leverage term is included for models DIAG.g, CCC.g, or PRCOMP.g.

n.predict: the number of steps ahead predicted from the current time.

cccor.choice: for ccc type models only, an integer specifying the correlation estimation:

0 - use sample correlation matrix and perform no further MLE estimation of the correlation matrix;

1 - use sample correlation matrix as the initial values and estimate the correlation matrix via MLE;

2 - use user supplied correlation matrix as the initial values and estimate the correlation matrix via MLE.

cccor.value: specifies the correlation matrix supplied by the user for cccor.choice=2.

cond.dist: conditional distribution given the past. The default is "gaussian" for multivariate Gaussian. An alternative is "t" for the multivariate t-distribution.

dist.par: the parameter for the conditional distribution.

dist.est: logical flag: if TRUE, the parameters of the conditional distribution will be estimated. The default is FALSE.

trace: logical flag: if TRUE, step-by-step optimization results will be printed on the screen, otherwise see opt.index for convergence information. The default is TRUE.

armaType: parameter that determines the type of ARMA coefficient matrix. Valid choices are "diag" for diagonal matrix, "full" for a full matrix, and "lower" for a lower triangular matrix.

control: a list of control parameters to be used in the numerical algorithms. See bhhh.control for the possible control parameters and their default settings.

algorithm: parameter that determines the algorithm used for maximum likelihood estimation. Currently only "bhhh" is available.

VALUE

an S version 3 object of class "mgarch", including the following components:

coef: the estimated coefficients in the model.

likelihood: the log-likelihood value at the optimum.

residuals: a numeric matrix, or a "timeSeries" object, which represents residuals from the fitted multivariate model.

R.t: an array of dimension (n, m, m), with m being the dimension of the model and n being the length of the series. This represents the estimated sequence of conditional correlation matrix.

S.t: an array of dimension (n, m, m), with m being the dimension of the model and n being the length of the series. This represents the estimated sequence of conditional covariance matrix.

sigma.t: a numeric matrix, or a "timeSeries" object, which represents the estimated conditional standard deviation sequence.

series: a numeric matrix, or a "timeSeries" object, which represents the original multivariate time series.

model: the model list of the fitted model.

opt.index: convergence indicator:

1 indicating BHHH convergence has been reached.

2 indicating the likelihood value has converged.

3 indicating a local maximum has been reached.

4 indicating the maximum number of iterations has been reached.

REFERENCES

Bollerslev, T., Engle, R. F., and Nelson, D. B. (1994). ARCH models. In *Handbook of Econometrics, Vol. IV*, Engle and McFadden, eds., Elsevces Science B.V. pp. 2959-3038,

Ding, Z. (1994). Time Series Analysis of Speculative Returns. *Ph.D. dissertation*, Dept. of Economics, University of California, San Diego.

SEE ALSO

bhhh.control, fgarch , garch .

EXAMPLE

```
hp.ibm = seriesMerge(hp.s, ibm.s)
hp.ibm.mod = mgarch(hp.ibm~1, ~dvec(1,1))
```

mimic Factor Mimicking Portfolios

DESCRIPTION
> Returns the weights for factor mimicking portfolios from a fitted multi-factor model.

```
mimic(object)
```

REQUIRED ARGUMENTS

`object` : an object of class `"mfactor"`, which is usually returned by a call to the `mfactor` function.

VALUE
> an object of class `"mimic"` which is a matrix with dimension `n` x `k` with the weights for factors in columns, where `n` is the number of variables and `k` is the number of factors.

DETAILS
> The function solves the linear system `RW = F` for the weights of factor mimicking portfolios, where `R` is the original data, and `F` are the factor returns. When the number of observations is smaller than the number of variables, `ginverse` is used to solve the system.

REFERENCES
> Grinold, R. C., and Kahn, R. N. (1999). *Active Portfolio Management*. McGraw-Hill.

SEE ALSO
> `ginverse`, `plot.summary.mimic`, `summary.mimic`.

EXAMPLE
```
folio.mf = mfactor(folio.dat, 15)
summary(mimic(folio.mf))
```

mk.fwd1 McCulloch and Kwon U.S. Term Structure Data

SUMMARY

The data sets `mk.fwd1`, `mk.fwd2`, `mk.par1`, `mk.par2`, `mk.zero1`, and `mk.zero2` are monthly `"timeSeries"` objects, which extend the McCulloch U.S. Treasury term structure data (appearing in the Handbook of Monetary Economics) to February 1991.

`mk.fwd1:` a `"timeSeries"` object with a 465 x 55 matrix in the data slot, representing the forward interest rate curve from December 1947 to August 1985.

`mk.fwd2:` a `"timeSeries"` object with a 67 x 55 matrix in the data slot, representing the forward interest rate curve from August 1985 to February 1991.

`mk.par1:` a `"timeSeries"` object with a 465 x 55 matrix in the data slot, representing the par-bond yield curve from December 1947 to August 1985.

`mk.par2:` a `"timeSeries"` object with a 67 x 55 matrix in the data slot, representing the par-bond yield curve from August 1985 to February 1991.

`mk.zero1:` a `"timeSeries"` object with a 465 x 55 matrix in the data slot, representing the zero-coupon yield curve from December 1947 to August 1985.

`mk.zero2:` a `"timeSeries"` object with a 67 x 55 matrix in the data slot, representing the zero-coupon yield curve from August 1985 to February 1991.

`mk.maturity:` a numeric vector of length 55, giving the fifty-five maturities in terms of years for the term structure.

SOURCE

McCulloch, J. H. (1990). U.S. term structure data: 1946-87. *Handbook of Monetary Economics*, B. M. Friedman and F. H. Hahn (eds.), Elsevier Science.

McCulloch, J. H., and Kwon, H.-C. (1993). U.S. term structure data: 1947-1991. Working Paper #93-6, Department of Economics, Ohio State University.

msft.dat Daily Microsoft Stock Prices

SUMMARY

This is a daily `"timeSeries"` object representing the open, high, low, close and volume information for Microsoft stocks from September 27, 2000 to September 27, 2001. The data has the following columns:

`Open`: the opening price.

`High`: the highest price in the day.

`Low`: the lowest price in the day.

`Close`: the closing price.

`Volume`: the trading volume.

ndx.dat Daily NADSAQ 100 Index

SUMMARY

This is a daily `"timeSeries"` object representing the open, high, low and close prices for NASDAQ 100 index from January 2, 1996 to October 12, 2001. The data has the following columns:

Open: the opening price.

High: the highest price in the day.

Low: the lowest price in the day.

Close: the closing price.

nelson.dat Extended Nelson-Plosser U.S. Macroeconomic Data

SUMMARY

This is an annual `"timeSeries"` object from 1860 to 1988 with fourteen columns (note that some earlier observations are missing). The original Nelson and Plosser (1982) data set runs from 1860 to 1970. It is extended to 1988 by Schotman and van Dijk (1991).

IP: industrial production index.

CPI: consumer price index.

VEL: velocity of money.

SP500: stock prices measured by Standard and Poor 500 index.

EMP: total employment.

UNEMP: total unemployment.

GNPD: GNP deflator.

M: money stock measured by M2.

BND: bond yield measured by basic yields of 30-year corporate bonds.

RPCGNP: real per capita GNP.

NGNP: nominal GNP.

RGNP: real GNP.

WG: nominal wages.

RWG: real wages.

SOURCE

Nelson, C. R., and Plosser, C. I. (1982). Trends and random walks in macroeconomic time series: some evidence and implications. *Journal of Monetary Economics*, 10:139-162.

Schotman, P. C., and van Dijk, H. K. (1991). On Bayesian route to unit roots. *Journal of Applied Econometrics*, 6:387-401.

newtaxes.dat Monthly Import Taxes of Argentina

SUMMARY
> The `newtaxes.dat` data frame has 10 rows and 1 column. The sample runs from January to October 1992.

DATA DESCRIPTION
> This data frame contains the following column:

`taxes:` monthly import taxes of Argentina.

nidd.annual The River Nidd Data

SUMMARY

This is a numeric vector of annual maximal levels of the River Nidd in Yorkshire. These data are suitable for analysis using `gev` function.

nidd.thresh The River Nidd Data

SUMMARY

This is a numeric vector of high river levels of the River Nidd in Yorkshire above a threshold value of 65. These data are suitable for analysis using gpd function.

nile.dat Annual Flow Volume from the River Nile

> This is a `"timeSeries"` object which consists of a series of readings of the annual flow volume of Nile at Aswan from 1871 to 1970.

REFERENCES

> Balke, N. S. (1993). Detecting level shifts in time series. *Journal of Business and Economic Statistics*, 11:81-92.
>
> Cobb, G. W. (1978). The problem of the Nile: conditional solution to a change point problem. *Biometrika*, 65:243-251.
>
> Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

NLSUR.object Nonlinear Seemingly Unrelated Regressions Model Objects

DESCRIPTION

These are objects of class `"NLSUR"` which inherit from the class `"SUR"` and represent the fit of a nonlinear seemingly unrelated regressions model.

GENERATION

This class of objects is returned from the `NLSUR` function.

METHODS

The `"NLSUR"` class of objects currently has methods for the following generic functions:

`coef`, `residuals`, `fitted`, `vcov`, `plot`, `print`, `summary`.

STRUCTURE

The following components must be present in a legitimate `"NLSUR"` object:

`call` : an image of the call that produced the object.

`objective` : the final value of the objective (sum of squares).

`message` : a statement of the reason for termination. For the possible messages and their meanings, please see the help file for `nlregb`.

`grad.norm` : the final norm of the objective gradient.

`iterations` : the total number of iterations before termination.

`r.evals` : the total number of residual evaluations before termination.

`j.evals` : the total number of jacobian evaluations before termination.

`scale` : the final value of the scale vector.

`coef` : a numeric vector giving all the estimated parameters of the nonlinear model.

`residuals` : a rectangular object with each column containing the residuals from each equation.

`fitted` : a rectangular object with each column containing the fitted values from each equation.

`cov` : the covariance matrix of the estimated model parameters.

`Sigma` : a matrix giving the residual covariance estimate.

`parm.list` : a list with length equal to the number of equations in the model, where each component of the list contains the parameter names for each equation.

`X.k` : a list with length equal to the number of equations in the model, where each component of the list contains the number of parameters in each equation.

SEE ALSO

NLSUR,nlregb .

NLSUR Fit a Nonlinear Seemingly Unrelated Regressions Model

DESCRIPTION

Fits a nonlinear seemingly unrelated regressions model to multivariate data.

```
NLSUR(formulas, data, na.rm=F, coef=NULL, start=NULL, end=NULL,
    control=NULL, ...)
```

REQUIRED ARGUMENTS

formulas : a list of `formula` objects, with each representing one nonlinear equation in the multivariate system, and with all the parameters explicitly named. In particular, any variables that are not present in `data` will be treated as a parameter.

data : a `data.frame` or a `"timeSeries"` object with a `data.frame` in the data slot. This will be used to evaluate the variables specified in `formulas`.

OPTIONAL ARGUMENTS

na.rm : a logical flag: if `TRUE`, missing values will be removed before fitting the model. The default is `FALSE`.

coef : a numeric vector giving the starting values of model parameters used by the nonlinear optimization algorithm. It should have names corresponding to the parameter names specified in `formulas`. By default, all the starting values are set to `0.1`.

start : a character string which can be passed to `timeDate` function to specify the starting date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

end : a character string which can be passed to `timeDate` function to specify the ending date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

control : a list of parameters by which the user can control various aspects of the nonlinear optimization algorithm. For details, see the help file for `nlregb.control`.

... : any optional arguments that can be passed to the `timeDate` function to parse the `start` or `end` specification.

VALUE

an object of class `"NLSUR"` which represents the fit of the nonlinear seemingly unrelated regressions model. See `NLSUR.object` for details.

DETAILS

The `NLSUR` function tries to estimate parameters for the nonlinear model that minimize the sum of squared residuals. In particular, the optimization algorithm underlying `nlregb` is used.

REFERENCES

Chambers, J. M., and Hastie, T.J. (Eds.) (1992). *Statistical Models in S*. Pacific Grove, CA.: Wadsworth and Brooks/Cole.

Gallant, A. R. (1975). Seemingly unrelated nonlinear regressions. *Journal of Econometrics*, 3:35-50.

SEE ALSO

NLSUR.object, SUR ,nlregb ,nlregb.control .

EXAMPLE

```
# simulate a multivariate data set.
set.seed(10)
x <- abs(rnorm(100))
sim.dat <- data.frame(x=x,
                      y1=1+2*x+rnorm(100),
                      y2=4+5*x^2+rnorm(100))
# estimate the nonlinear model
NLSUR(list(y1~c1+c2*x^c3, y2~c4+c5*x^c6), data=sim.dat)
```

normal.copula.object Normal Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"normal.copula"` , which inherits from `copula` class.

It is the copula for the bivariate Gaussian distribution.

GENERATION

This class of objects is constructed from the `normal.copula(delta)` function.

METHODS

The `"normal.copula"` class of objects currently has methods for the following generic functions:

`pcopula, dcopula, rcopula, contour.plot, Kendalls.tau, Spearmans.rho`.

Furthermore, following functions are implemented for this class: `persp.dcopula, persp.pcopula, contour.dcopula, contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"normal.copula"` class object:

`parameters` : a numerical value for the parameter `delta` (between 0 and 1).

`param.names` : `delta`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (Normal copula family).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`copula.object`.

normal.mix.copula.object Mixture Normal Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"normal.mix.copula"` , which inherits from `copula` class.

It is the copula for the mixture of two Gaussian distribution.

GENERATION

This class of objects is constructed from the `normal.mix.copula(p, delta1, delta2)` function.

METHODS

The `"normal.mix.copula"` class of objects currently has methods for the following generic functions:

`pcopula`, `dcopula`, `rcopula`, `contour.plot`, `Kendalls.tau`, `Spearmans.rho`.

Furthermore, following functions are implemented for this class: `persp.dcopula`, `persp.pcopula`, `contour.dcopula`, `contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"normal.mix.copula"` class object:

`parameters` : a vector of values for the parameter(s) `p` (between 0 and 1), `delta1` (between 0 and 1), and `delta2` (between 0 and 1).

`param.names` : `p`, `delta1`, `delta2`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (Mixed Normal copula family).

SEE ALSO

`normal.copula.object`,`copula.object` .

normalTest Test for Univariate Normality

DESCRIPTION

Returns the statistics for various tests for univariate normality.

This function is an S Version 3 generic (see `Methods`); method functions can be written to handle specific classes of data. If the input is a data object, then the normality test is applied on the data set directly; if the input is a fitted model object, then the test is applied on the `"residuals"` component of the object if it is present.

```
normalTest(x, method="sw", na.rm=F)
```

REQUIRED ARGUMENTS

`x` : a vector, matrix, data frame, or a `"timeSeries"` object with a numeric object in the data slot, or a list with a `"residuals"` component such as most fitted model objects.

OPTIONAL ARGUMENTS

`method` : a character string specifying the type of the test. Currently the valid choices are: `"sw"` for Shapiro-Wilks test, and `"jb"` for Jarque-Bera test. The default is `"sw"`.

`na.rm` : a logical flag: if `TRUE`, missing values will be removed before computing the tests. The default is `FALSE`.

VALUE

an object of class `"normalTest"`, which contains the following components:

`method` : a character string which is the same as in the input.

`stat` : the test statistics according to the chosen `method`.

`distribution` : a character string giving the distribution of the test statistic under the null hypothesis.

`parameters` : the degree of freedom of `distribution` under the null hypothesis.

`p.value` : the probability that the null hypothesis is true.

`n` : an integer giving the sample size.

`na` : an integer giving the number of missing values in the input if any.

REFERENCES

Jarque, C. M., and Bera, A. K. (1987). A test for normality of observations and regression residuals. *International Statistical Review*, 55(5):163-172.

Royston, J. P. (1982). An extension of Shapiro and Wilk's W test for normality to large samples. *Applied Statistics*, 31:115-124.

SEE ALSO

`archTest,autocorTest` .

EXAMPLE

`normalTest(dell.s)`

nyse Value Weighted Market Returns

SUMMARY

The `nyse` data vector is a data set with 7420 values, representing the daily value weighted market returns from July 3, 1962, to December 31, 1991 (7420 trading days). The time series `nyse.s` is the last 2000 data points from `nyse`, representing the daily value weighted market returns from Feburary 2, 1984, to December 31, 1991.

OLS.fit.qr   Fit an Ordinary Least Squares Model

DESCRIPTION

Returns an `"OLS"` object representing the ordinary least squares fit. It is called by the function `OLS` , and its direct use is not recommended.

```
OLS.fit.qr(x, y)
```

REQUIRED ARGUMENTS

x : a numeric vector or matrix for the predictors in the ordinary least squares estimation. Typically, but not necessarily, `x` will be the model matrix generated by `OLS` function.

y : a numeric vector or matrix for the response in the ordinary least squares estimation.

VALUE

an object of class `"OLS"` representing the fit. See `OLS.object` for details.

SEE ALSO

`lm.fit.qr`, `OLS` , `OLS.object` .

OLS.object   Ordinary Least Squares Model Object

DESCRIPTION

These are objects of class `"OLS"`. They represent the fit of an ordinary least squares estimation.

GENERATION

This class of objects is returned from the `OLS` function.

METHODS

The `"OLS"` class of objects currently has methods for the following generic functions:

`coef, residuals, fitted, vcov, formula, plot, predict, print, summary`.

STRUCTURE

The following components must be present in a legitimate `"OLS"` object:

`call` : an image of the call that produced the object, but with the arguments all named and with the actual formula included as the `formula` argument.

`coef` : the coefficients of the ordinary least squares fit of the response to the columns of the model matrix.

`residuals` : the residuals from the fit. If weights were used, then the residuals are the raw residuals - the weights are not taken into account.

`fitted` : the fitted values from the fit. If weights were used, the fitted values are not adjusted for the weights.

`weights` : the weights used in the regression if present.

`df.resid` : the number of degrees of freedom for residuals.

`terms` : an object of mode `expression` and class `term` summarizing the formula. Used by various methods, but typically not of direct relevance to users.

`contrasts` : a list containing sufficient information to construct the contrasts used to fit any factors occurring in the model. The list contains entries that are either matrices or character vectors. When a factor is coded by contrasts, the corresponding contrast matrix is stored in this list. Factors that appear only as dummy variables and variables in the model that are matrices correspond to character vectors in the list. The character vector has the level names for a factor or the column labels for a matrix.

`assign` : the list of assignments of coefficients (and effects) to the terms in the model. The names of this list are the names of the terms. The `ith` element of the list is the vector saying which coefficients correspond to the `ith` term.

SEE ALSO

`lm`, `lm.object` , `OLS` .

OLS Ordinary Least Squares Estimation

DESCRIPTION

Returns an object of class `"OLS"` that represents an ordinary least squares fit.

```
OLS(formula, data, weights, subset, na.rm=F, contrasts=NULL,
    start=NULL, end=NULL, ...)
```

REQUIRED ARGUMENTS

`formula` : a `formula` object, with the response on the left of a ∼ operator, and the terms, separated by + operators, on the right. The variables cannot be `"timeSeries"` objects if `data` argument is not specified.

OPTIONAL ARGUMENTS

`data` : a data frame or `"timeSeries"` data frame in which to interpret the variables named in the `formula` and `subset` arguments. If `data` is missing, the variables in the model formula should be in the search path, and cannot be `"timeSeries"` objects.

`weights` : a vector of observation weights; if supplied, the algorithm fits to minimize the sum of the weights multiplied into the squared residuals. The length of weights must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous, compared to use of the `subset` argument.

`subset` : an expression specifying which subset of observations should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), a numeric vector indicating the observation numbers to be included, or a character vector of the observation names that should be included. All observations are included by default.

`na.rm` : a logical flag: if `TRUE`, missing values will be removed before fitting the model. The default is `FALSE`.

`contrasts` : a list giving contrasts for some or all of the factors appearing in the model formula. An element in the list should have the same name as the factor variable it encodes, and it should be either a contrast matrix (any full-rank matrix with as many rows as there are levels in the factor), or a function that computes such a matrix given the number of levels.

start : a character string which can be passed to `timeDate` function to specify the starting date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

end : a character string which can be passed to `timeDate` function to specify the ending date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

... : any optional arguments that can be passed to the `timeDate` function to parse the `start` or `end` specification.

VALUE

an object of class `"OLS"` representing the fit. See `OLS.object` for details.

DETAILS

The function `OLS` presents an alternative to `lm` with support for some popular econometric features, such as adjusted R-squared, Durbin-Watson statistic, heteroskedasticity-consistent covariance matrix, dynamic regressions, etc. See `OLS.object` for more details.

If the response variable is multivariate, then a list of `"OLS"` objects is returned.

SEE ALSO

`lm`, `OLS.object` ,`RLS` , `rollOLS` .

EXAMPLE

```
OLS(Oil~Market, data=oilcity)

OLS(SP500~IP, data=nelson.dat, start="01/01/1951")
```

outliers.object Outliers Objects Extracted from `arima.rob` Objects

DESCRIPTION

These are objects of class `"outliers"` extracted an object of class `"arima.rob"`, which contain information about the detected outliers (and level shifts).

GENERATION

This class of objects is returned from the `outliers` function.

METHODS

`print`, `summary` .

STRUCTURE

The following components must be included in a legitimate `"outliers"` object:

VALUE

`nout:` the number of outliers (and level shifts) detected.

`outlier.index:` the index of each detected outlier (or level shift).

`outlier.type:` the type of each detected outlier (or level shift): 1=innovation outlier, 2=additive outlier, 3=level shift.

`outlier.impact:` the size of each detected outlier (or level shift).

`outlier.t.statistics:` the t-statistics for each detected outlier (or level shift).

`outlier.positions:` the `"timeDate"` objects associated with the detected outliers if the original data is a `"timeSeries"` object.

`sigma0:` the estimate of the innovation scale before correcting the outliers.

`sigma:` the estimate of the innovation scale after correcting the outliers.

`ierror:` the error indicator: if `ierror > 0` the search for outliers was stopped, the algorithm detected too many outliers.

SEE ALSO

`outliersarima.rob,arima.rob.object` .

outliers Outliers Extraction for an `arima.rob` Object

DESCRIPTION

Returns an object of class `"outliers"`.

```
outliers(object, iter=NULL)
```

REQUIRED ARGUMENTS

`object`: an object of class `"arima.rob"`.

OPTIONAL ARGUMENTS

`iter`: a number specifying from which iteration to extract the detected out-
liers, if the `iter` argument passed to the function `arima.rob` that
produced `object` is non-zero. The default is set to `NULL`.

VALUE

an object of class `"outliers"`. If `iter` is `NULL`, the object contains all
the detected outliers (and level shifts). If `iter` is not `NULL`, the object
contains the outliers (and level shifts) detected in iteration `iter`. See
`outliers.object` for components of the returned object.

SEE ALSO

`outliers.object`,`arima.rob` ,`arima.rob.object` .

EXAMPLE

```
frip.rr = arima.rob(log(frip)~1, p=2, d=1,iter=2)
frip.outliers.all = outliers(frip.rr)
frip.outliers.2 = outliers(frip.rr,iter=2)
```

pbivd Density, CDF and Random Generation from a Bivariate Distribution

DESCRIPTION

Computes density, cumulative probability and generate a bivariate distribution with arbitrary marginals.

```
pbivd(dist, x, y)
dbivd(dist, x, y)
rbivd(dist, n)
```

REQUIRED ARGUMENTS

dist : an object of class "bivd".

  x : a numerical vector of quantiles in the first variable.

  y : a numerical vector of quantiles in the second variable.

  n : a numerical value of desired size of the random sample.

VALUE

Probability, density and a sample generated from a particular bivariate distribution.

DETAILS

These functions are implemented for all "bivd" sub-classes. The function rbivd causes the creation of the dataset .Random.seed if it does not already exist, otherwise its value is updated.

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

bivd.object, pcopula .

EXAMPLE

```
bivd1 <- bivd(normal.copula(0.6), "norm", "t", param.marginX = c(0,2.5),
param.marginY = 5)
plot(rbivd(bivd1))
```

pcoint   Cumulative Probability of Cointegration Distributions

DESCRIPTION

Returns cumulative probability of asymptotic or finite sample distribution of cointegration test statistics.

```
pcoint(q, n.sample=0, n.series=2, estimator="c",
       statistic="t")
```

REQUIRED ARGUMENTS

q : vector of quantiles or test statistics. Missing values (NA) are allowed.

OPTIONAL ARGUMENTS

n.sample : the number of observations in the sample from which the test statistics are computed. Specify n.sample=0 for asymptotic cumulative probabilities. The default is 0.

n.series : the number of series used in the cointegration test. The default is 2.

estimator : a character string describing the regression from which the test statistics are computed. Valid choices are: "nc" for a regression with no intercept (constant) nor time trend, and "c" for a regression with an intercept (constant) but no time trend, "ct" for a regression with an intercept (constant) and a time trend. The default is "c".

statistic : a character string describing the type of test statistic. Valid choices are "t" for t-statistic, and "n" for normalized statistic (sometimes referred to as the rho-statistic). The default is "t".

VALUE

vector of cumulative probabilities from the specified distribution.

DETAILS

The functions pcoint and qcoint only apply to the cointegration tests computed using the Engle-Granger procedure. They are based on the response surface regression coefficients provided by MacKinnon (1996) and do not apply to Johansen's rank tests.

REFERENCES

MacKinnon, J. G. (1996). Numerical distribution functions for unit root and cointegration tests. *Journal of Applied Econometrics*, 11:601-618.

SEE ALSO

coint, qcoint ,punitroot ,qunitroot .

EXAMPLE

```
# finite sample cumulative probability of n-statistic
pcoint(1.2836, n.sample=100, n.series=3, estimator="nc", statistic="n")
```

pcopula Density, Cumulative Probability and Random Generation from Copulas

DESCRIPTION

Computes density, cumulative probability of a copula and generates two random variables with uniform marginals and joint cumulative distribution given by a copula object.

```
pcopula(copula, u,v)
dcopula(copula, u, v)
rcopula(copula, n)
```

REQUIRED ARGUMENTS

copula: an object of class "copula".

u: a numerical vector of quantiles in the first variable.

v: a numerical vector of quantiles in the second variable.

n: a numerical value of desired size of the random sample.

VALUE

Probability, density and a sample generated from a particular copula.

DETAILS

This is a generic function. All copula classes have their method functions (empirical copula only has pcopula implemented.) The function rcopula causes the creation of the dataset .Random.seed if it does not already exist, otherwise its value is updated.

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

copula.object, pbivd .

EXAMPLE

```
# Creates an image plot of the joint density of two r.v.s with
# standard Gaussian marginals and Gumbel copula.
b1c = gumbel.copula(1.5)
x = c(-150:150)/50
y = x
x1 = matrix(x, nrow=length(x), ncol=length(x), byrow=T)
y1 = matrix(y, nrow=length(x), ncol=length(x), byrow=F)
z = dcopula(b1c, pnorm(x1), pnorm(y1)) * dnorm(x1) * dnorm(y1)
z = matrix(z, ncol=length(x), nrow=length(x), byrow=T)
image(x,y,z)
```

```
# Now generate a sample of size 2000 and
# plot in on top of the image
simdata = rcopula(b1c, 2000)
Xsim = qnorm(simdata$x)
Ysim = qnorm(simdata$y)
points(Xsim, Ysim, cex = 0.5)
```

pdl Polynomial Distributed Lags

DESCRIPTION
Returns a regressor matrix suitable for polynomial distributed lags.

```
pdl(x, d=2, q=3, trim=F)
```

REQUIRED ARGUMENTS

x : a vector, a matrix, or a `"timeSeries"` object. Missing values (`NA`) are allowed.

OPTIONAL ARGUMENTS

d : an integer specifying the order of the polynomial. The default is 2.

q : an integer specifying the number of lags to use in creating polynomial distributed lags. This must be greater than `d`. The default is set to 3.

trim : a logical flag: if `TRUE`, the missing values at the beginning of the returned matrix will be trimmed. The default is `FALSE`.

VALUE
a matrix representing the regressor matrix.

SEE ALSO
`tslag`.

EXAMPLE
```
pdl(stack.loss, d=2, p=3)
```

pgev Generalized Extreme Value Distribution

DESCRIPTION

Cumulative probability, quantiles, density and random generation from the generalized extreme value distribution.


```
pgev(q, xi=1, mu=0, sigma=1)
qgev(p, xi=1, mu=0, sigma=1)
dgev(x, xi=1, mu=0, sigma=1)
rgev(n, xi=1, mu=0, sigma=1)
```

REQUIRED ARGUMENTS

q: vector of quantiles.

p: vector of probabilities.

x: vector of values at which to evaluate density.

n: sample size.

OPTIONAL ARGUMENTS

xi: a numeric value specifying the shape parameter of a GEV distribution. It is replicated to be the same length as p or q or the number of deviates generated.

mu: a numeric value specifying the location parameter of a GEV distribution. It is replicated to be the same length as p or q or the number of deviates generated.

sigma: a numeric value specifying the scale parameter of a GEV distribution. It is replicated to be the same length as p or q or the number of deviates generated.

VALUE

Cumulative probability (pgev), quantile (qgev), density (dgev) or random sample (rgev) for the GEV distribution with shape, location and scale parameters of xi, mu and sigma. The function rgev causes creation of the dataset .Random.seed if it does not already exist, otherwise its value is updated.

DETAILS

Elements of q or q that are missing will cause the corresponding elements of the result to be missing.

The ratio of uniform deviates is used by rgev to generate GEV deviates.

SEE ALSO

gev,pgpd .

EXAMPLE

```
# sample of 20 with xi=1, mu=0, and sigma=1
rgev(20)
```

pgpd Generalized Pareto Distribution

DESCRIPTION

      Cumulative probability, quantiles, density and random generation from
      the generalized Pareto distribution.


```
pgpd(q, xi, mu=0, beta=1)
qgpd(p, xi, mu=0, beta=1)
dgpd(x, xi, mu=0, beta=1)
rgpd(n, xi, mu=0, beta=1)
```

REQUIRED ARGUMENTS

      `q`: vector of quantiles.

      `p`: vector of probabilities.

      `x`: vector of values at which to evaluate density.

      `n` : sample size.

      `xi`: shape parameter of a GPD distribution. It is replicated to be the same
      length as `p` or `q` or the number of deviates generated.

OPTIONAL ARGUMENTS

      `mu`: a numeric value specifying the location parameter of a GPD distribu-
      tion. It is replicated to be the same length as `p` or `q` or the number of
      deviates generated.

      `beta`: a numeric value specifying the scale parameter of a GPD distribution.
      It is replicated to be the same length as `p` or `q` or the number of
      deviates generated.

VALUE

      Cumulative probability (`pgpd`), quantile (`qgpd`), density (`dgpd`) or ran-
      dom sample (`rgpd`) for the GPD distribution with shape, location and
      scale parameters of `xi`, `mu` and `beta`. The function `rgpd` causes cre-
      ation of the dataset `.Random.seed` if it does not already exist, other-
      wise its value is updated.

DETAILS

      Elements of `q` or `q` that are missing will cause the corresponding ele-
      ments of the result to be missing.

      The ratio of uniform deviates is used by `rgpd` to generate GPD devi-
      ates.

SEE ALSO

gpd,pgev .

EXAMPLE

```
# sample of 20 with xi=1, mu=0, and beta=1
rgev(20, 1)
```

plot.arima.rob Generate Diagnostic Plots for an `arima.rob` Object

DESCRIPTION

Creates a set of plots suitable for assessing the fit of a robust RE-GARIMA model of class `"arima.rob"`, along with plots of the detected outliers (and level shifts).

```
plot.arima.rob(object)
```

REQUIRED ARGUMENTS

`object`: an object of class `"arima.rob"`.

This function displays a menu listing all the plots that can be produced, which include:

autocorrelations of the innovations

partial autocorrelations of the innovations

normal QQ-plot of the innovations

detected outliers

DETAILS

This function is a method for the generic function `plot` for class `"arima.rob"` . It can be invoked by calling `plot` for an object of the appropriate class, or directly by calling `plot.arima.rob` regardless of the class of the object.

plot.compare.mgarch Plot Multivariate GARCH Models Comparison

DESCRIPTION

Plots output from a GARCH models comparison. Typically this function is called through the generic function `plot` .

```
plot.compare.mgarch(x, qq=F, hgrid=F, vgrid=F, lag.max=NULL,
    ci=2, ...)
```

REQUIRED ARGUMENTS

x: an object inheriting from class `"compare.mgarch"`. Typically this is the result of a call to the function `compare.mgarch`.

OPTIONAL ARGUMENTS

qq: a logical flag: if `TRUE` quantile-quantile plots will be drawn; if `FALSE` autocorrelation function plots will be drawn.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

lag.max: an integer specifying the maximum lag steps in ACF plot. The default is set to `10*log10(n)` where `n` is the number of observations in the original series.

ci: the factor that determines the width of the confidence interval in terms of the conditional standard deviation for choice 2. For example, `ci=2` indicates that a confidence interval of 2 standard deviations wide will be put around the estimated conditional mean. The default is set to 2.

...: additional graphics parameters may be supplied.

Creates a plot on the current graphics device. For each component series of each model a quantile-quantile or an autocorrelation function plot of the squared residuals is drawn.

SEE ALSO

`compare.mgarch`.

plot.FARIMA Generate Diagnostic Plots for a Fractional ARIMA Object

DESCRIPTION

Creates a set of plots suitable for assessing a fitted fractional ARIMA model of class `"FARIMA"`.

```
plot.FARIMA(x, ask=T, id.n=3, which.plots=NULL,
            n.density=50, hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

x : an object inheriting from class `"FARIMA"`.

OPTIONAL ARGUMENTS

ask : a logical flag: if `TRUE`, `plot.FARIMA` operates in interactive mode. The default is `TRUE`

id.n : number of points (must be less than the number of observations) to be identified in the appropriate plots. These will be the `id.n` largest points in absolute value. Set to `FALSE` if no identified points are desired. The default is 3.

which.plots : numeric vector taking values between 1 and 10 inclusive, specifying which of the plots (described below) to display.

n.density : an integer specifying the number of classes (i.e., bars) the histogram should have. The default is 50.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

... : any optional argument that may be passed down to the plot function.

An appropriate `x-y` plot is produced to display diagnostic plots. These can be one or all of the following choices:

Response versus Fitted Values

Response and Fitted Values

Normal QQplot of Residuals

Residuals

Standardized Residuals

Residual Histogram

Residual ACF

Residual PACF

Residual^2 ACF

Residual^2 PACF

DETAILS

This function is a method for the generic function `plot` for class `"FARIMA"`. It can be invoked by calling `plot` for an object `x` of the appropriate class, or directly by calling `plot.FARIMA` regardless of the class of the object.

When `ask=T`, rather than produce each plot sequentially, `plot.FARIMA` displays a menu listing all the plots that can be produced. If the menu is not desired but a pause between plots is still wanted one must set `par(ask=T)` before invoking this command with argument `ask=F`.

SEE ALSO

`FARIMA.object,plot` .

EXAMPLE

```
dell.d = FARIMA(dell.s, p=0, q=0)
plot(dell.d)
```

plot.forecast Plot of Predictions and Forecasts

DESCRIPTION

Creates a plot of the predictions or forecasts along with the in-sample observations.

```
plot.forecast(x, xold, n.old=NULL, width=1, xlab="index",
              hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

   x : an object inheriting from class `"forecast"`.

xold : the original data used to fit the model.

OPTIONAL ARGUMENTS

n.old : a positive integer that is smaller than the number of observations used to fit the model. Only the last `n.old` observations in the original data will be plotted together with the forecasts. If `n.old=NULL`, all the observations will be plotted. The default is `NULL`.

width : a number that specifies the width of the confidence bands around the forecasts, in units of standard errors. The default is 1.

 xlab : a character string that specifies the text to be used as the label for x-axis. The default is `"index"`.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

  ... : any optional argument that may be passed down to the plot function.

A plot of the predictions/forecasts will be shown in a graphical device together with the original data used to fit the model. A confidence band will also be shown if the standard errors of the forecasts are present.

DETAILS

This function is a method for the generic function `plot` for class `"forecast"`. It can be invoked by calling `plot` for an object `x` of the appropriate class, or directly by calling `plot.forecast` regardless of the class of the object.

SEE ALSO

`plot`.

EXAMPLE

```
pol.mod = VAR(cbind(CP,M2,GDP,CPI)~ar(13), data = log(policy.dat))
pol.forecast = predict(pol.mod, 12)
plot(pol.forecast, log(policy.dat[,c("CP","M2","GDP", "CPI")]),
     n.old=20, hgrid=T, vgrid=T)
```

plot.garch Plot Fitted GARCH Model

DESCRIPTION

The plot method `plot.garch` provides 12 different types of plots. The user may select from a menu which displays the plot types on the screen. See the examples below.

```
plot.garch(x, ask=T, which.plots=NULL, hgrid=F, vgrid=F,
        lag.max=NULL, ci=2, id.n=3, ...)
```

REQUIRED ARGUMENTS

x: an object of class `"garch"`, usually generated by the function `garch`.

OPTIONAL ARGUMENTS

ask: a logical flag: if `TRUE`, `plot.garch` acts in an interactive mode.

which.plots: numeric vector taking values between 1 and 12 inclusive, specifying which of the plots (described below) to display.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

lag.max: an integer specifying the maximum lag steps in ACF plot. The default is set to `10*log10(n)` where `n` is the number of observations in the original series.

ci: the factor that determines the width of the confidence interval in terms of the conditional standard deviation for choice 2. For example, `ci=2` indicates that a confidence interval of 2 standard deviations wide will be put around the estimated conditional mean. The default is set to 2.

id.n: an integer specifying the number of points to be identified in the QQ-plot.

Time series plots of the residuals, standardized residuals, conditional standardized deviations, together with the ACF plots with the maximum lag steps specified by `lag.max` and the QQ plot of the standardized residuals.

SEE ALSO

`plot.mgarch`.

EXAMPLE

```
hp.mod = garch(hp.s~ar(1), ~garch(1,1))
plot(hp.mod)

Make a plot selection (or 0 to exit):
1: plot: All
2: plot: Series and Conditional SD
3: plot: Series with 2 Conditional SD Superimposed
4: plot: ACF of the Observations
5: plot: ACF of Squared Observations
6: plot: Cross Corr. between Squared Series and Series
7: plot: Residuals
8: plot: Conditional Standard Deviation
9: plot: Standardized Residuals
10: plot: ACF of Standardized Residuals
11: plot: ACF of Squared Standardized Residuals
12: plot: Cross Corr. between Squared Std.Res and Std.Res
13: plot: QQ-plot of Standardized Residuals
```

plot.gev Plot Fitted GEV Model

DESCRIPTION

Provides two different residual plots for assessing fitted GEV model. The user selects the plot type from a menu. See the examples below.

```
plot.gev(x, ...)
```

REQUIRED ARGUMENTS

x: an object of class `"gev"`.

DETAILS

Data are converted to unit exponentially distributed residuals under null hypothesis that GEV fits. Two diagnostics for iid exponential data are offered. A scatter plot or a QQ-plot against exponential distribution is drawn on a graphical device.

SEE ALSO

gev.

EXAMPLE

```
out = gev(bmw,"month")
plot(out)

Make a plot selection (or 0 to exit):
1: plot: Scatterplot of Residuals
2: plot: QQplot of Residuals
```

plot.gpdbiv Plot Fitted Bivariate GPD Model

DESCRIPTION

Provides five plots summarizing a fitted bivariate GPD model. The user selects the plot type from a menu. See the examples below.

```
plot.gpdbiv(out, extend=1.1, n.contours=15)
```

REQUIRED ARGUMENTS

out: a "gpdbiv" object as returned by the gpdbiv function.

OPTIONAL ARGUMENTS

extend: a numeric value specifying how far x-axis should extend as a multiple of the largest data value. The default is 1.1.

n.contours: an integer specifying the number of contours in bivariate contour plots. The default is 15.

DETAILS

Option 1 plots the threshold exceedance data; option 2 plots contours of the fitted bivariate distribution function in the joint upper tail (above both thresholds); option 3 plots corresponding contours of the fitted joint survival function; plots 4 and 5 show the fitted tails of the marginal distributions. Plots are drawn on a graphical device.

SEE ALSO

gpdbiv, interpret.gpdbiv , gpd , plot.gpd , tailplot .

EXAMPLE

```
out = gpdbiv(-bmw, -siemens, ne1=100, ne2=100)
plot(out)
Make a plot selection (or 0 to exit):
1: plot: Exceedance data
2: plot: Contours of Bivariate Distribution Function
3: plot: Contours of Bivariate Survival Function
4: plot: Tail of Marginal 1
5: plot: Tail of Marginal 2
```

plot.gpd Plot Fitted GPD Model

DESCRIPTION

Provides four different plots for assessing a fitted GPD model. The user selects the plot type from a menu. See the examples below.

```
plot.gpd(gpd.obj, optlog=NA, extend=1, labels=T, ...)
```

REQUIRED ARGUMENTS

`gpd.obj`: a "gpd" object. This is usually returned by the `gpd` function.

OPTIONAL ARGUMENTS

`optlog`: a character string giving a particular choice of logarithmic axes (for plots 1 and 2): `"x"` for x-axis only; `"y"` y-axis only; `"xy"` for both axes; `""` for neither axis. The default is `NA` which generates logarithmic x-axis for plot 1 and both axes on logarithmic scale for plot 2.

`extend`: a numerical value greater than 1 specifying how far x-axis should extend as a multiple of the largest data value (for plots 1 and 2 only). This argument is useful for showing estimated quantiles beyond data. The default is `1.5`.

`labels`: a logical flag for plots 1 and 2 specifying whether or not axes should be labeled. The default is `TRUE`.

Plots 1 and 2 cause the creation of an object called `lastcurve` in frame 0 containing details of the plot. This is needed if quantile estimates are to be added.

DETAILS

For plots 3 and 4 data are converted to unit exponentially distributed residuals under null hypothesis that GEV fits. The function `qplot` is used in plot 4.

SEE ALSO

`gpd`, `shape` , `tailplot` , `quant` .

EXAMPLE

```
out = gpd(danish, 10)
plot(out)

Make a plot selection (or 0 to exit):
1: plot: Excess Distribution
2: plot: Tail of Underlying Distribution
3: plot: Scatterplot of Residuals
4: plot: QQplot of Residuals
```

plot.impDecomp Plot of Impulse Response Function or Forecast Error
Variance Decomposition

DESCRIPTION

Creates a plot of the impulse response function, or forecast error vari-
ance decomposition of a VAR model.

```
plot.impDecomp(x, width=1, hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

x : an object inheriting from class `"impDecomp"`, usually returned by
`impRes` or `fevDec`.

OPTIONAL ARGUMENTS

width : a number that specifies the width of the confidence bands around the
values, in units of standard errors. This is only used if the standard
errors were computed in `impRes` or `fevDec`. The default is 1.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The
default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The
default is `FALSE`.

... : any optional argument that may be passed down to the plot function.

VALUE

an object of class `trellis`, which is automatically plotted by `print.trellis`.
A plot of either the impulse response function or forecast error vari-
ance decomposition will be shown on a graphics device.

DETAILS

This function is a method for the generic function `plot` for class
`"impDecomp"`. It can be invoked by calling `plot` for an object `x` of
the appropriate class, or directly by calling `plot.impDecomp` regard-
less of the class of the object.

SEE ALSO

fevDec, impRes , plot .

EXAMPLE

```
pol.mod = VAR(cbind(M2,GDP,U)~ar(13), data=log(policy.dat))
pol.fev = fevDec(pol.mod, period=5)
plot(pol.fev, hgrid=T, vgrid=T)
```

plot.KalmanFil Generate Diagnostic Plots for a `KalmanFil` Object

DESCRIPTION

Generates a set of plots suitable for assessing a Kalman filtering object.

```
plot.KalmanFil(x, ask=T, which.plots=NULL, nclass="Sturges",
    lag.max=20, id.n=3, hgrid=F, vgrid=F, one.plot=T, ...)
```

REQUIRED ARGUMENTS

x : an object of class `"KalmanFil"` as returned by `KalmanFil` function.

OPTIONAL ARGUMENTS

ask : a logical flag: if `TRUE`, `plot.KalmanFil` operates in interactive mode. The default is `TRUE`.

which.plots : numeric vector taking values between 1 and 5 inclusive, specifying which of the plots (described below) to display.

nclass : recommendation for the number of classes (i.e., bars) the histogram should have. This may be an integer, a function which returns an integer, or a character string specifying which built-in method to use. Available methods for calculating the number of classes are Sturges (`"Sturges"`), Freedman-Diaconis (`"FD"`), and Scott (`"Scott"`). The default is `"Sturges"`.

lag.max : a positive integer that specifies the maximum number of lags to use to compute the autocorrelation function. The default is 20.

id.n : number of points (must be less than the number of observations) to be identified in the appropriate plots. These will be the `id.n` largest points in absolute value. Set to `FALSE` if no identified points are desired. The default is 3.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the plots. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the plots. The default is `FALSE`.

one.plot : a logical flag: if `TRUE`, the innovations for all the variables will be drawn on one plot. For multivariate models, it is usually better to set `one.plot=F`, which will draw each innovation in its own panel. The default is `TRUE`.

... : any optional arguments that may be passed down to the plot function.

An appropriate `x-y` plot is produced to display diagnostic plots. These can be one or all of the following choices:

1: innovations;

2: standardized innovations;

3: innovation histogram;

4: normal QQ-plot of innovations;

5: innovation ACF.

SEE ALSO

`acf`,`hist` ,`histPlot` ,`qqPlot` .

plot.KalmanSmo Plot of Standardized Smoothing Residuals

DESCRIPTION

Generates a trellis display of standardized smoothing residuals of a "KalmanSmo" object.

```
plot.KalmanSmo(x, hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

x : an object of class "KalmanSmo" as returned by a call to KalmanSmo function.

OPTIONAL ARGUMENTS

hgrid : a logical flag: if TRUE, horizontal grids will be drawn on the plots. The default is FALSE.

vgrid : a logical flag: if TRUE, vertical grids will be drawn on the plots. The default is FALSE.

... : any optional arguments that may be passed down to the plot function.

VALUE

an object of class "trellis", which is automatically plotted by print.trellis.

plot.mfactor   Generate Diagnostic Plots for a Multi-Factor Model Object

DESCRIPTION

Creates a set of plots suitable for assessing a multi-factor model of class `"mfactor"`.

```
plot.mfactor(x, variables, cumulative=T, style="bar",
             ask=T, which.plots=NULL, hgrid=F, vgrid=F,
             ...)
```

REQUIRED ARGUMENTS

`x` : an object inheriting from class `"mfactor"`.

OPTIONAL ARGUMENTS

`variables` : an integer vector telling which variables are to be plotted. The default is to plot all the variables, or the number of variables explaining 90% of the variance, whichever is bigger.

`cumulative` : a logical flag: if `TRUE`, the cumulative fraction of the variance is printed above each bar in the plot.

`ask` : a logical flag: if `TRUE`, `plot.mfactor` operates in interactive mode. The default is `TRUE`

`which.plots` : numeric vector taking values between 1 and 2 inclusive, specifying which of the plots (described below) to display.

`hgrid` : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

`vgrid` : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

`...` : any optional argument that may be passed down to the plot function.

An appropriate plot is produced to display the multi-factor model. These can be one or all of the following choices:

Screeplot of Eigenvalues

Factor Returns

DETAILS

This function is a method for the generic function `plot` for class `"mfactor"`. It can be invoked by calling `plot` for an object x of the appropriate class, or directly by calling `plot.mfactor` regardless of the class of the object.

When `ask=T`, rather than produce each plot sequentially, `plot.mfactor` displays a menu listing all the plots that can be produced. If the menu is not desired but a pause between plots is still wanted one must set `par(ask=T)` before invoking this command with argument `ask=F`.

SEE ALSO

`fplot,plot ,screeplot.mfactor` .

EXAMPLE

```
folio.mf = mfactor(folio.dat, 15)
plot(folio.mf)
```

plot.mgarch Plot Fitted Multivariate GARCH Model

DESCRIPTION

The plot method `plot.mgarch` provides 10 different types of plots. The user may select from a menu which displays the plot types.

```
plot.mgarch(x, ask=T, which.plots=NULL, hgrid=F, vgrid=F,
    lag.max=NULL, id.n=3, ...)
```

REQUIRED ARGUMENTS

x: an object of class `"mgarch"`, usually generated by the function `mgarch`.

OPTIONAL ARGUMENTS

ask: a logical flag: if `TRUE`, `plot.mgarch` acts in an interactive mode.

which.plots: numeric vector taking values between 1 and 12 inclusive, specifying which of the plots (described below) to display.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

lag.max: an integer specifying the maximum lag steps in ACF plot. The default is set to `10*log10(n)` where `n` is the number of observations in the original series.

id.n: an integer specifying the number of points to be identified in the QQ-plot.

Time series plots of the residuals, standardized residuals, conditional standardized deviations, together with the ACF plots with the maximum lag steps specified by `lag.max` and the QQ plot of the standardized residuals.

EXAMPLE

```
hp.ibm = seriesMerge(hp.s, ibm.s)
hp.ibm.mod = mgarch(hp.ibm~1, ~dvec(1,1))
plot(hp.ibm.mod, ask=T)
Make a plot selection (or 0 to exit):
1: plot: All
2: plot: Original Observations
3: plot: ACF of Observations
4: plot: ACF of Squared Observations
```

```
 5: plot: Residuals
 6: plot: Conditional SD
 7: plot: Standardized Residual
 8: plot: ACF of Standardized Residuals
 9: plot: ACF of Squared Std. Residuals
10: plot: QQ-Plots of Standardized Residuals
```

plot.OLS Generate Diagnostic Plots for an OLS Object

DESCRIPTION

Creates a set of plots suitable for assessing a fitted ordinary least squares model of class `"OLS"`.

```
plot.OLS(x, ask=T, which.plots=NULL, n.density=50,
        hgrid=F, vgrid=F, id.n=3, ...)
```

REQUIRED ARGUMENTS

x : an object inheriting from class `"OLS"`.

OPTIONAL ARGUMENTS

ask : a logical flag: if `TRUE`, `plot.OLS` operates in interactive mode. The default is `TRUE`

id.n : number of points (must be less than the number of observations) to be identified in the appropriate plots. These will be the `id.n` largest points in absolute value. Set to `FALSE` if no identified points are desired. The default is 3.

which.plots : numeric vector taking values between 1 and 10 inclusive, specifying which of the plots (described below) to display.

n.density : an integer specifying the number of classes (i.e., bars) the histogram should have. The default is 50.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

... : any optional argument that may be passed down to the plot function.

An appropriate x-y plot is produced to display diagnostic plots. These can be one or all of the following choices:

Response versus Fitted Values

Response and Fitted Values

Normal QQplot of Residuals

Residuals

Standardized Residuals

Residual Histogram

Residual ACF

Residual PACF

Residual^2 ACF

Residual^2 PACF

DETAILS

This function is a method for the generic function `plot` for class `"OLS"`. It can be invoked by calling `plot` for an object x of the appropriate class, or directly by calling `plot.OLS` regardless of the class of the object.

When `ask=T`, rather than produce each plot sequentially, `plot.OLS` displays a menu listing all the plots that can be produced. If the menu is not desired but a pause between plots is still wanted one must set `par(ask=T)` before invoking this command with argument `ask=F`.

SEE ALSO

`plot`.

EXAMPLE

```
stack.dat = data.frame(Loss=stack.loss, stack.x)
stack.OLS = OLS(Loss~Air.Flow+Water.Temp+Acid.Conc., data=stack.dat)
plot(stack.OLS)
```

PlotPos.LMOM Plotting Position Estimate of the Sample L-Moments

DESCRIPTION

Computes plotting position estimate of the sample L-Moments.

```
PlotPos.LMOM(sample, gamma=-0.35, delta=0)
```

REQUIRED ARGUMENTS

`sample` : a numeric vector of data points or `"timeSeries"` object. Missing values (NAs) are not allowed.

`gamma` : a numerical value specifying plotting position estimate of `gamma`. The default is `-0.35`.

`delta` : a numerical value specifying plotting position estimate of `delta`. The default is `0`.

VALUE

a vector of length four with the estimates `l1`, `l2`, `tau3`, and `tau4`, based on the plotting position.

DETAILS

`PlotPos.LMOM` is named `plotting.position.estimator.LMOM` in the original EVANESCE library.

REFERENCES

Hosking, J. R. M. (1986). The theory of probability weighted moments, Research Report RC12210, IBM Research, Yorktown Heights, NY.

Hosking, J. R. M. (1990). L-moments: analysis and estimation of distributions using linear combinations of order statistics, *Journal of Royal Statistical Society*, 52(1):105-124.

SEE ALSO

`sample.LMOM`.

EXAMPLE

```
PlotPos.LMOM(rgpd(10, beta=3, mu=0.6, xi=0.5))
```

plot.pot Plot Fitted POT Model

DESCRIPTION

> Provides seven different plots for assessing a fitted POT model. The user selects the plot type from a menu. See the examples below.

> ```
> plot.pot(pot.obj)
> ```

REQUIRED ARGUMENTS

`pot.obj`: a "pot" object as returned by the `pot` function.

DETAILS

> Plot 1 displays the exceedance process of the chosen threshold. Plots 2-4 assess the Poisson nature of the exceedance process by looking at the scaled gaps between exceedances, which should be iid unit exponentially distributed. Plots 5-6 assess the GPD nature of the excesses by looking at suitably defined residuals, which should again be iid unit exponentially distributed. Option 8 allows the user to call GPD plotting functions.

SEE ALSO

> `pot`, `gpd` , `plot.gpd` .

EXAMPLE

> ```
> out = pot(danish,10)
> plot(out)
>
> Make a plot selection (or 0 to exit):
> 1: plot: Point Process of Exceedances
> 2: plot: Scatterplot of Gaps
> 3: plot: Qplot of Gaps
> 4: plot: ACF of Gaps
> 5: plot: Scatterplot of Residuals
> 6: plot: Qplot of Residuals
> 7: plot: ACF of Residuals
> 8: plot: Go to GPD Plots
> ```

plot.predict.garch Plot Predicted GARCH Object

DESCRIPTION

Plots the predicted GARCH objects.

```
plot.predict.garch(x, sigma.pred=T, series.pred=F, hgrid=F,
    vgrid=F, sigma.lab=NULL, series.lab=NULL, ...)
```

REQUIRED ARGUMENTS

x: an object of class "predict.garch".

OPTIONAL ARGUMENTS

sigma.pred: a logical flag: if TRUE, plot the predicted conditional standard deviation.

series.pred: a logical flag: if TRUE, plot the predicted conditional mean.

hgrid : a logical flag: if TRUE, horizontal grids will be drawn on the plot. The default is FALSE.

vgrid : a logical flag: if TRUE, vertical grids will be drawn on the plot. The default is FALSE.

series.lab: a character string that specifies the label for the conditional mean plot.

sigma.lab: a character string that specifies the label for the conditional standard deviation plot.

The predicted conditional mean and standard deviation are plotted on the current graphics device.

SEE ALSO

plot.predict.mgarch, predict.garch , predict.mgarch .

plot.predict.mgarch Plot Predicted Multivariate GARCH Object

DESCRIPTION

Plots the predicted multivariate GARCH objects.

```
plot.predict.mgarch(x, ask=T, which.plots=NULL, hgrid=F,
      vgrid=F, portfolio=F, weights=NULL, ...)
```

REQUIRED ARGUMENTS

x: an object of class "predict.mgarch".

OPTIONAL ARGUMENTS

ask: a logical flag: if TRUE, plot.mgarch acts in an interactive mode.

which.plots: numeric vector taking values between 1 and 12 inclusive, specifying which of the plots (described below) to display.

hgrid : a logical flag: if TRUE, horizontal grids will be drawn on the plot. The default is FALSE.

vgrid : a logical flag: if TRUE, vertical grids will be drawn on the plot. The default is FALSE.

portfolio: a logical flag: if TRUE, a portfolio will be formed. The default is FALSE.

weights: a numeric vector specifying the weights to form the portfolio. This must be supplied if portfolio=T. The default is NULL.

The predicted conditional mean and standard deviation series are plotted on the current graphics device.

SEE ALSO

plot.predict.garch, predict.mgarch , predict.garch .

plot.RLS Generate Diagnostic Plots for a RLS Object

DESCRIPTION

Creates a set of plots suitable for assessing a fitted recursive least squares model of class `"RLS"`.

```
plot.RLS(x, ask=T, which.plots=NULL, hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

`x` : an object inheriting from class `"RLS"`.

OPTIONAL ARGUMENTS

`ask` : a logical flag: if `TRUE`, `plot.RLS` operates in interactive mode. The default is `TRUE`

`which.plots` : numeric vector taking values between 1 and 3 inclusive, specifying which of the plots (described below) to display.

`hgrid` : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

`vgrid` : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

`...` : any optional argument that may be passed down to the plot function.

An appropriate x-y plot is produced to display diagnostic plots. These can be one or all of the following choices:

Coefficient Estimates

CUSUM of Residuals

CUSUM of Squared Residuals

DETAILS

This function is a method for the generic function `plot` for class `"RLS"`. It can be invoked by calling `plot` for an object `x` of the appropriate class, or directly by calling `plot.RLS` regardless of the class of the object.

When `ask=T`, rather than produce each plot sequentially, `plot.RLS` displays a menu listing all the plots that can be produced. If the menu is not desired but a pause between plots is still wanted one must set `par(ask=T)` before invoking this command with argument `ask=F`.

SEE ALSO

`RLS.object`,`plot` .

EXAMPLE

```
stack.dat = data.frame(Loss=stack.loss, stack.x)
stack.RLS = RLS(Loss~Air.Flow+Water.Temp+Acid.Conc., data=stack.dat)
plot(stack.RLS)
```

plot.rollOLS Generate Diagnostic Plots for a Rolling OLS Object

DESCRIPTION

Creates a set of plots suitable for assessing a fitted rolling OLS model of class `"rollOLS"`.

```
plot.rollOLS(x, ask=T, which.plots=NULL, hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

`x` : an object inheriting from class `"rollOLS"`.

OPTIONAL ARGUMENTS

`ask` : a logical flag: if `TRUE`, `plot.rollOLS` operates in interactive mode. The default is `TRUE`

`which.plots` : numeric vector taking values between 1 and 3 inclusive, specifying which of the plots (described below) to display.

`hgrid` : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

`vgrid` : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

`...` : any optional argument that may be passed down to the plot function.

An appropriate `x-y` plot is produced to display diagnostic plots. These can be one or all of the following choices:

Coef Estimates

Coef Estimates with Confidence Intervals

Residual Scale Estimates

DETAILS

This function is a method for the generic function `plot` for class `"rollOLS"`. It can be invoked by calling `plot` for an object `x` of the appropriate class, or directly by calling `plot.rollOLS` regardless of the class of the object.

When `ask=T`, rather than produce each plot sequentially, `plot.rollOLS` displays a menu listing all the plots that can be produced. If the menu is not desired but a pause between plots is still wanted one must set `par(ask=T)` before invoking this command with argument `ask=F`.

SEE ALSO

`rollOLS.object`,`plot` .

EXAMPLE

```
stack.dat = data.frame(Loss=stack.loss, stack.x)
stack.r = rollOLS(Loss~Water.Temp, data=stack.dat, width=6, incr=2)
plot(stack.r)
```

plot.SEMIFAR Generate Diagnostic Plots for a Semiparametric FAR
Object

DESCRIPTION

Creates a set of plots suitable for assessing a fitted semiparametric
fractional AR model of class "SEMIFAR".

```
plot.SEMIFAR(x, ask=T, id.n=3, which.plots=NULL, hgrid=F,
             vgrid=F, series=NULL, n.density=50, ...)
```

REQUIRED ARGUMENTS

x : an object inheriting from class "SEMIFAR".

OPTIONAL ARGUMENTS

ask : a logical flag: if TRUE, plot.SEMIFAR operates in interactive mode.
The default is TRUE

id.n : number of points (must be less than the number of observations) to
be identified in the appropriate plots. These will be the id.n largest
points in absolute value. Set to FALSE if no identified points are de-
sired. The default is 3.

which.plots : numeric vector taking values between 1 and 8 inclusive, speci-
fying which of the plots (described below) to display.

n.density : an integer specifying the number of classes (i.e., bars) the his-
togram should have. The default is 50.

series : the original time series data used to fit the SEMIFAR model. This
can be given if the original data is not accessible from the evaluating
frame.

hgrid : a logical flag: if TRUE, horizontal grids will be drawn on the plot. The
default is FALSE.

vgrid : a logical flag: if TRUE, vertical grids will be drawn on the plot. The
default is FALSE.

... : any optional argument that may be passed down to the plot function.

An appropriate x-y plot is produced to display diagnostic plots. These
can be one or all of the following choices:

trend, fitted values, and residuals

normal QQ-plot of residuals

standardized residuals

residual histogram

residual ACF

residual PACF

residual^2 ACF

residual^2 PACF

DETAILS

This function is a method for the generic function `plot` for class
`"SEMIFAR"`. It can be invoked by calling `plot` for an object `x` of the
appropriate class, or directly by calling `plot.SEMIFAR` regardless of
the class of the object.

When `ask=T`, rather than produce each plot sequentially, `plot.SEMIFAR`
displays a menu listing all the plots that can be produced. If the menu
is not desired but a pause between plots is still wanted one must set
`par(ask=T)` before invoking this command with argument `ask=F`.

SEE ALSO

`SEMIFAR.object`,`plot` .

EXAMPLE

```
dell.d = SEMIAR(dell.s)
plot(dell.d)
```

plot.SsfMomentEst Plots of Smoothed States or Disturbance

DESCRIPTION

Generates a trellis display of the filtered/smoothed state variables, response variables, or disturbance.

```
plot.SsfMomentEst(x, hgrid=F, vgrid=F, ...)
plot.SsfCondDens(x, hgrid=F, vgrid=F, ...)
plot.SimSmoDraw(x, hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

x : an object of class `"SsfMomentEst"`, `"SsfCondDens"`, or `"SimSmoDraw"`.

OPTIONAL ARGUMENTS

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the plots. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the plots. The default is `FALSE`.

... : any optional arguments that may be passed down to the plot function.

VALUE

an object of class `"trellis"`, which is automatically plotted by `print.trellis`.

plot.summary.mimic Plot of Top Positions in Factor Mimicking Portfolios

DESCRIPTION

Creates a bar plot of top positions in factor mimicking portfolios.


```
plot.summary.mimic(x, ...)
```

REQUIRED ARGUMENTS

x : an object inheriting from class `"summary.mimic"`.

OPTIONAL ARGUMENTS

... : any optional argument that may be passed down to the plot function.

A bar plot of the top positions in each factor mimicking portfolio will be drawn in a graphical device.

DETAILS

This function is a method for the generic function `plot` for class `"summary.mimic"`. It can be invoked by calling `plot` for an object x of the appropriate class, or directly by calling `plot.summary.mimic` regardless of the class of the object.

SEE ALSO

`plot`,`summary.mimic` .

EXAMPLE

```
folio.mf = mfactor(folio.dat, 15)
folio.s = summary(mimic(folio.mf))
plot(folio.s)
```

plot.term.struct Plot of Fitted Term Structure of Interest Rates

DESCRIPTION

Creates a plot of the fitted term structure of interest rates.

```
plot.term.struct(x, n.points=100, hgrid=F, vgrid=F,
                 strip.text=NULL, ...)
```

REQUIRED ARGUMENTS

x : an object inheriting from class `"term.struct"`, usually returned by the `term.struct` function.

OPTIONAL ARGUMENTS

n.points : an integer giving the number of points to use in the plot.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

strip.text : a character string which will appear in the strip of the trellis plot. The default is `""`, which draws nothing in the strip.

... : any optional argument that may be passed down to the plot function.

VALUE

an object of class `trellis`, which is automatically plotted by `print.trellis`. A plot of the term structure of interest rates will be shown on a graphical device.

DETAILS

This function is a method for the generic function `plot` for class `"term.struct"`. It can be invoked by calling `plot` for an object x of the appropriate class, or directly by calling `plot.term.struct` regardless of the class of the object.

SEE ALSO

`term.struct`, `plot` .

EXAMPLE

```
ts.mod = term.struct(mk.zero2[21,], mk.maturity, na.rm=T, method="ns",
            input="spot", plot=F)
plot(ts.mod)
```

plot.VAR Generate Diagnostic Plots for a VAR Object

DESCRIPTION

Creates a set of plots suitable for assessing a fitted VAR model of class `"VAR"`.

```
plot.VAR(x, ask=T, which.plots=NULL, hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

`x` : an object inheriting from class `"VAR"`.

OPTIONAL ARGUMENTS

`ask` : a logical flag: if `TRUE`, `plot.VAR` operates in interactive mode. The default is `TRUE`

`which.plots` : numeric vector taking values between 1 and 7 inclusive, specifying which of the plots (described below) to display.

`hgrid` : a logical flag: if `TRUE`, horizontal grids will be drawn on the plot. The default is `FALSE`.

`vgrid` : a logical flag: if `TRUE`, vertical grids will be drawn on the plot. The default is `FALSE`.

`...` : any optional argument that may be passed down to the plot function.

An appropriate `x-y` plot is produced to display diagnostic plots. These can be one or all of the following choices:

Response versus Fitted Values

Residuals

Normal QQplot of Residuals

Residual ACF

Residual PACF

Residual^2 ACF

Residual^2 PACF

DETAILS

This function is a method for the generic function `plot` for class `"VAR"`. It can be invoked by calling `plot` for an object `x` of the appropriate class, or directly by calling `plot.VAR` regardless of the class of the object.

When `ask=T`, rather than produce each plot sequentially, `plot.VAR` displays a menu listing all the plots that can be produced. If the menu is not desired but a pause between plots is still wanted one must set `par(ask=T)` before invoking this command with argument `ask=F`.

SEE ALSO

VAR,plot .

EXAMPLE

```
pol.mod = VAR(cbind(CP,GDP,U)~ar(6), data=log(policy.dat))
plot(pol.mod)
```

plot.VECM Generate Diagnostic Plots for a VECM Object

DESCRIPTION

Creates a set of plots suitable for assessing a fitted VECM model of class "VECM".

```
plot.VECM(x, ask=T, which.plots=NULL, hgrid=F, vgrid=F, ...)
```

REQUIRED ARGUMENTS

x : an object inheriting from class "VECM".

OPTIONAL ARGUMENTS

ask : a logical flag: if TRUE, plot.VECM operates in interactive mode. The default is TRUE

which.plots : numeric vector taking values between 1 and 12 inclusive, specifying which of the plots (described below) to display.

hgrid : a logical flag: if TRUE, horizontal grids will be drawn on the plot. The default is FALSE.

vgrid : a logical flag: if TRUE, vertical grids will be drawn on the plot. The default is FALSE.

... : any optional argument that may be passed down to the plot function.

An appropriate x-y plot is produced to display diagnostic plots. These can be one or all of the following choices:

Response versus Fitted Values

Residuals

Normal QQplot of Residuals

Residual ACF

Residual PACF

Residual^2 ACF

Residual^2 PACF

Cointegrating Residuals

ACF of Cointegrating Residuals

PACF of Cointegrating Residuals

ACF of Squared Cointegrating Residuals

PACF of Squared Cointegrating Residuals

DETAILS

This function is a method for the generic function `plot` for class
"VECM". It can be invoked by calling `plot` for an object `x` of the
appropriate class, or directly by calling `plot.VECM` regardless of the
class of the object.

When `ask=T`, rather than produce each plot sequentially, `plot.VECM`
displays a menu listing all the plots that can be produced. If the menu
is not desired but a pause between plots is still wanted one must set
`par(ask=T)` before invoking this command with argument `ask=F`.

SEE ALSO

VECM,VECM.object ,plot .

EXAMPLE

```
mk.coint = coint(mk.zero2[,1:5], lags=1, trend="rc")
mk.vecm = VECM(mk.coint, coint.rank=3)
plot(mk.vecm)
```

policy.dat Monthly U.S. Macroeconomic Policy Data

SUMMARY

This is a monthly `"timeSeries"` object with dimension 471 x 6, which runs from January 1959 to March 1998.

CP: International Monetary Fund's index of world commodity prices.

M2: seasonally adjusted M2 money stock in billions of dollars.

FFR: federal funds rate.

GDP: seasonally adjusted real GDP in billions of 1992 dollars.

CPI: seasonally adjusted consumer price index for urban consumers.

U: seasonally adjusted civilian unemployment rate.

SOURCE

Zha, T. (1998). A dynamic multivariate model for use in formulating policy. *Federal Reserve Bank of Atlanta Economic Review*, First Quarter, 1998.

pot.object POT (Peak Over Threshold) Model Object

DESCRIPTION

These are objects of class `"pot"`. They represent the fit of a POT model for excesses over a high threshold.

GENERATION

This class of objects is returned from the `pot` function.

METHODS

The `"pot"` class of objects currently has methods for the following generic functions:

`plot`.

STRUCTURE

The following components must be present in a legitimate `"pot"` object:

`n` : the length of the series data.

`period` : the start and end time position (or index) of the time series data (or signal series data).

`data` : the exceedances series data.

`span` : the time span (or number) of the series data.

`threshold` : the threshold.

`p.less.thresh` : the percentage of the data less than the threshold.

`n.exceed` : the length of exceedances.

`run` : the run length parameter if decluster is performed.

`par.ests` : the parameter estimates vector.

`par.ses` : the standard error of parameter estimates.

`varcov` : the variance-covariance matrix of parameter estimates.

`intensity` : the average rate at which excesses occur.

`nllh.final` : the maximum negative log likelihood function value.

`converged` : the logical flag indicating if ML estimation converged.

`note` : the string character describing the type of the convergence.

SEE ALSO

`pot`.

pot Fit Peaks Over Thresholds Model

DESCRIPTION

Fits a Poisson point process to the data, an approach sometimes known as peaks over thresholds (POT).

```
pot(data, threshold=NA, nextremes=NA, run=NA, plot=T)
```

REQUIRED ARGUMENTS

data: a numeric vector or `"timeSeries"` object.

threshold: a threshold value. Either `threshold` or `nextremes` must be given but not both.

nextremes: the number of upper extremes to be used. Either `nextremes` or `threshold` must be given but not both.

run: if the data are to be declustered the run length parameter for the runs method should be supplied.

plot: a logical flag controlling whether or not a picture should be drawn if declustering is performed. The default is `TRUE`.

VALUE

an object of class `"pot"` representing the fit. See `pot.object` for details. A plot of the exceedances series data, a `qplot` of the gaps, a plot of the declustered series data, and a `qplot` of gaps of the declustered data will be drawn on a graphical device if `plot=T`.

SEE ALSO

`decluster`, `plot.pot` , `gpd` , `plot.gpd` .

EXAMPLE

```
# Fits POT model to Danish fire insurance losses
out = pot(danish, 10)
```

predict.arima.rob Use `predict()` on an `arima.rob` Object

DESCRIPTION

Predicts from a fitted `"arima.rob"` object.

```
predict.arima.rob(object, n.predict=1, newdata=NULL,
        olddata=NULL, se.fit=F)
```

REQUIRED ARGUMENTS

`object`: an object of class `"arima.rob"`.

OPTIONAL ARGUMENTS

`n.predict`: the number of predictions to be returned.

`newdata`: a data frame containing the future values of exogenous variables, if any, at which predictions are required.

`olddata` : a data frame containing the original data used to fit `object`. This is only required if `tslag` is used to create distributed lags of exogenous variables in the original call that generated `object`.

`se.fit`: logical flag: if `TRUE`, pointwise standard errors are computed along with the predictions.

VALUE

an object of class `"forecast"`. See `forecast.object` for details.

DETAILS

This function is a method for the generic function `predict` for class `"arima.rob"` . It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.arima.rob` regardless of the class of the object.

SEE ALSO

`arima.rob`,`forecast.object` .

EXAMPLE

```
import.rr = arimar.rob(import~taxes-1, data=import.dat, p=2, d=1)
import.hat = predict(import.rr, 5, newdata=newtaxes.dat, se=T)
```

predict.BVAR Use `predict()` on a `BVAR` Object

DESCRIPTION

Performs prediction from a fitted Bayesian VAR model.

```
predict.BVAR(object, n.predict=1, newdata=NULL, olddata=NULL,
             seed=100, n.sim=1000)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"BVAR"`.

OPTIONAL ARGUMENTS

`n.predict` : an integer specifying the number of periods to predict ahead.

`newdata` : a data frame containing the future values of exogenous variables, if any, at which predictions are required.

`olddata` : a data frame containing the original data used to fit `object`. This is only required if `tslag` is used to create distributed lags of exogenous variables in the original call that generated `object`.

`seed` : an integer between 0 and 1023, or a previous value of `.Random.seed`, which represents the random seed to be used.

`n.sim` : an integer specifying the number of simulations.

VALUE

an object of class `"forecast"`. The `draws` element of the object contains all the simulated draws. See `forecast.object` for details. Sets the value of the `.Random.seed` object in the working directory.

DETAILS

This function is a method for the generic function `predict` for class `"BVAR"`. It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.BVAR` regardless of the class of the object.

REFERENCES

Sims, C. A., and Zha, T. (1998). Bayesian methods for dynamic multivariate models. *International Economic Review*, 39(4):949-968.

SEE ALSO

`cpredict`, `forecast.object` , `predict` .

EXAMPLE

```
pol.mod = BVAR(cbind(CP,M2,GDP,CPI)~ar(13), data = log(policy.dat),
               unit.root=T, coint=T)
pol.forecast = predict(pol.mod, 12)
```

predict.FARIMA Use `predict()` on a `FARIMA` Object

DESCRIPTION
Performs prediction from a fitted fractional ARIMA model.

```
predict.FARIMA(object, n.predict=1, series=NULL,
               ar.approx=50, kapprox=100000)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"FARIMA"`.

OPTIONAL ARGUMENTS

`n.predict` : an integer specifying the number of periods to predict ahead.

`series` : the original time series data used to fit the FARIMA model. This can be given if the original data is not accessible from the evaluating frame.

`ar.approx` : an integer specifying the number of lagged values to use for prediction. The default is 50.

`kapprox` : a large integer that determines how many Fourier frequencies to use in evaluating the theoretical spectrum of a FARIMA model. The default is set to `100000`.

VALUE
an object of class `"forecast"`. See `forecast.object` for details.

DETAILS
This function is a method for the generic function `predict` for class `"FARIMA"`. It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.FARIMA` regardless of the class of the object.

REFERENCES
Bhansali, R. J., and Kokoszka, P. S. (2001). Computation of the forecast coefficients for multistep prediction of long-range dependent time series. *International Journal of Forecasting*, forthcoming.

Brockwell, P. J., and David, R. A. (1991). *Time Series: Theory and Methods*, Springer-Verlag.

SEE ALSO
`forecast.object`, `predict` .

EXAMPLE
```
ts.sim = simulate.FARIMA(list(d=.3, ar=.5, ma=0.1, mean=1), n=3000)
f.mod = FARIMA(ts.sim, p=1, q=1, mmax=0)
predict(f.mod, 10, ar=100)
```

predict.fgarch Make Predictionns from a Fractionally Integrated GARCH
Model

DESCRIPTION

Predicts future realizations for the time series and conditional stan-
dard deviation series based on a fitted `fgarch` model and the data.

```
predict.fgarch(object, n.predict=1, n.lag=1000)
```

REQUIRED ARGUMENTS

`object`: an object of the fitted model of class `"fgarch"`.

OPTIONAL ARGUMENTS

`n.predict`: number of steps to be predicted ahead.

`n.lag`: lag truncations to be used to approximate fractional integartion.

VALUE

an object of class `"predict.fgarch"` containing the following two
components:

`series.pred`: the predicted time series from the current time up to `n.predict`
steps ahead.

`sigma.pred`: the predicted conditional standard deviation series.

SEE ALSO

`predict`, `predict.garch`, `plot.predict.garch`.

predict.fiegarch Predict from Fractionally Integrated GARCH Models

DESCRIPTION

Computes forecasts from a fractionally integrated GARCH models. It is called by `predict.fgarch` and not supposed to be called by the users directly.

```
predict.fiegarch(object, y, sigma, res, Cm, P, Q, q, ny,
                  n.predict, dphi, x=NULL, z=NULL)
predict.figarch(object, y, sigma, res, Cm, P, Q, p, ny,
                 n.predict, dphi, x=NULL, z=NULL)
```

predict.garch Make Predictions from a GARCH Model

DESCRIPTION

Predicts the future realizations for the time series and conditional standard deviation series based on a fitted `garch` model and the data.

```
predict.garch(object, n.predict=1)
```

REQUIRED ARGUMENTS

`object:` an object of the fitted model of class `"garch"`.

OPTIONAL ARGUMENTS

`n.predict:` number of steps to be predicted ahead.

VALUE

an object of class `"predict.garch"` containing the following three components:

`series.pred:` the predicted time series from the current time up to `n.predict` steps ahead.

`sigma.pred:` the predicted conditional standard deviation series.

`asymp.sd:` the asymptotic standard deviation.

SEE ALSO

`predict, plot.predict.garch`.

EXAMPLE

```
hp.fit = garch(hp.s~1, ~garch(1,1))
hp.new = predict(hp.fit, 3)
```

predict.mgarch Make Predictions from a Multivariate GARCH Model

DESCRIPTION

Predicts the future realizations for the time series and conditional standard deviation series based on a fitted `mgarch` model and the data.

```
predict.mgarch(object, n.predict=1)
```

REQUIRED ARGUMENTS

`object`: an object of the fitted model of class `"mgarch"`.

OPTIONAL ARGUMENTS

`n.predict`: number of steps to be predicted.

VALUE

an object of class `"predict.mgarch"` containing the following three components:

`series.pred`: the predicted time series from the current time up to `n.predict` steps ahead.

`sigma.pred`: the predicted conditional standard deviation series.

`R.pred`: the conditional correlation matrix series for the multivariate case.

SEE ALSO

`predict`, `plot.predict.mgarch`.

EXAMPLE

```
hp.ibm = seriesMerge(hp.s, ibm.s)
hp.ibm.fit = mgarch(hp.ibm~1, ~dvec(1,1))
hp.ibm.pred = predict(hp.ibm.fit, 3)
```

predict.OLS Use `predict()` on an `OLS` Object

DESCRIPTION

Performs prediction from a fitted OLS model.

```
predict.OLS(object, n.predict, newdata=NULL, olddata=NULL)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"OLS"`.

`n.predict` : an integer specifying the number of periods to predict ahead.

OPTIONAL ARGUMENTS

`newdata` : a data frame containing the future values of exogenous variables, if any, at which predictions are required.

`olddata` : a data frame containing the original data used to fit `object`. This is only required if `tslag` is used to create distributed lags of exogenous variables in the original call that generated `object`.

VALUE

an object of class `"forecast"`. See `forecast.object` for details.

DETAILS

This function is a method for the generic function `predict` for class `"OLS"`. It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.OLS` regardless of the class of the object.

SEE ALSO

`forecast.object`, `predict` .

EXAMPLE

```
# should return the same values as the fitted values
policy.m = OLS(log(CPI)~log(M2), data=policy.dat)
predict(policy.m, 10, newdata=policy.dat[1:10, "M2"])
```

predict.rollOLS Use `predict()` on a `rollOLS` Object

DESCRIPTION

Performs out-of-sample prediction from a fitted rolling OLS model.

```
predict.rollOLS(object, n.steps=1, ...)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"rollOLS"`.

OPTIONAL ARGUMENTS

`n.steps` : an integer vector specifying the number of periods to predict ahead.

VALUE

a list of out-of-sample predictions with length equal to the length of `n.steps`, with each component representing the corresponding predictions for all the rolling samples.

DETAILS

This function is a method for the generic function `predict` for class `"rollOLS"`. It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.rollOLS` regardless of the class of the object.

The prediction results can be easily used to compute root mean squared errors (RMSE) or other measures of predictive accuracy.

SEE ALSO

`predict`.

EXAMPLE

```
stack.dat = data.frame(Loss=stack.loss, stack.x)
stack.m = rollOLS(Loss~Water.Temp, data=stack.dat, width=6, incr=2)
predict(stack.m, c(1,2))
```

predict.SEMIFAR Use `predict()` on a `SEMIFAR` Object

DESCRIPTION

Performs prediction from a fitted SEMIFAR model.

```
predict.SEMIFAR(object, n.predict=1, series=NULL,
                ar.approx=50, kapprox=100000,
                trend="constant")
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"SEMIFAR"`.

OPTIONAL ARGUMENTS

`n.predict` : an integer specifying the number of periods to predict ahead.

`series` : the original time series data used to fit the SEMIFAR model. This can be given if the original data is not accessible from the evaluating frame.

`ar.approx` : an integer specifying the number of lagged values to use for prediction. The default is 50.

`kapprox` : a large integer that determines how many Fourier frequencies to use in evaluating the theoretical spectrum of a FARIMA model. The default is set to `100000`.

`trend` : a character string specifying the method for trend extrapolation. Valid choices are: `"constant"` for constant extrapolation, and `"linear"` for linear extrapolation. The default is `"constant"`.

VALUE

an object of class `"forecast"`. See `forecast.object` for details.

DETAILS

This function is a method for the generic function `predict` for class `"SEMIFAR"`. It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.SEMIFAR` regardless of the class of the object.

REFERENCES

Beran, J., and Ocker, D. (1999). SEMIFAR forecasts, with applications to foreign exchange rates. *Journal of Statistical Planning and Inference*, 80:137-153.

Bhansali, R. J., and Kokoszka, P. S. (2001). Computation of the forecast coefficients for multistep prediction of long-range dependent time series. *International Journal of Forecasting*, forthcoming.

Brockwell, P. J., and David, R. A. (1991). *Time Series: Theory and Methods*, Springer-Verlag.

SEE ALSO

`forecast.object`, `predict` .

EXAMPLE

```
ts.sim = simulate.FARIMA(list(d=.3, ar=.5, ma=0.1, mean=1), n=3000)
f.mod = SEMIFAR(ts.sim, mmax=0)
predict(f.mod, 10, ar=100)
```

predict.term.struct Use `predict()` on a `term.struct` Object

DESCRIPTION

Performs interpolation/extrapolation from a fitted term structure model.

```
predict.term.struct(x, maturity)
```

REQUIRED ARGUMENTS

x : an object inheriting from class `"term.struct"`. This is usually returned by a call to the `term.struct` function.

maturity : a numeric vector giving the maturities at which to interpolate or extrapolate the term structure. This is specified in terms of years.

VALUE

the interest rates corresponding to `maturity` according to the fitted term structure model.

DETAILS

This function is a method for the generic function `predict` for class `"term.struct"`. It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.term.struct` regardless of the class of the object.

This is called by `plot.term.struct` to plot the term structure.

SEE ALSO

`plot.term.struct`,`predict` .

EXAMPLE

```
yield.m = term.struct(mk.zero2[21,], mk.maturity, na.rm=T, method="ns",
            input="spot", plot=T)
predict(yield.m, mk.maturity)
```

predict.VAR Use `predict()` on a `VAR` Object

DESCRIPTION

Performs prediction from a fitted VAR model.

```
predict.VAR(object, n.predict=1, newdata=NULL, olddata=NULL,
            method="asymptotic", unbiased=T, fs.correction=F,
            seed=100, n.sim=1000)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"VAR"`.

OPTIONAL ARGUMENTS

`n.predict` : an integer specifying the number of periods to predict ahead.

`newdata` : a data frame containing the future values of exogenous variables, if any, at which predictions are required.

`olddata` : a data frame containing the original data used to fit `object`. This is only required if `tslag` is used to create distributed lags of exogenous variables in the original call that generated `object`.

`method` : a character string specifying the prediction method. Valid choices are: `"asymptotic"`, for which the forecasts are computed based on asymptotic theory; `"mc"`, for which the forecasts are computed using Monte Carlo simulation; and `"bootstrap"`, for which the forecasts are computed using bootstrap. The default is `"asymptotic"`.

`unbiased` : a logical flag; if `TRUE`, unbiased estimate of error covariance will be used; if `FALSE`, the maximum likelihood estimate of error covariance will be used. The default is `TRUE`.

`fs.correction` : a logical flag. This is only used if `method="asymptotic"`. If `TRUE`, a finite sample correction will be employed as described in Luktepohl (1990). The default is `FALSE`.

`seed` : an integer between 0 and 1023, or a previous value of `.Random.seed`, which represents the random seed to be used. This is only used if `method="mc"` or `method="bootstrap"`.

`n.sim` : an integer specifying the number of simulations. This is only used if `method="mc"` or `method="bootstrap"`.

VALUE

an object of class `"forecast"`. See `forecast.object` for details. Sets the value of the `.Random.seed` object in the working directory if `method="mc"` or `method="bootstrap"` .

DETAILS

> This function is a method for the generic function `predict` for class
> `"VAR"`. It can be invoked by calling `predict` for an object of the
> appropriate class, or directly by calling `predict.VAR` regardless of the
> class of the object.
>
> The asymptotic forecasts are computed as described in Lutkepohl
> (1990). If `method` is not `"asymptotic"`, either a Monte Carlo method
> or a bootstrap method can be used, but neither takes into account the
> parameter uncertainty. If parameter uncertainty needs to be consid-
> ered, `predict.BVAR` can be used instead.

REFERENCES

> Lutkepohl, H. (1990). *Introduction to Multiple Time Series Analysis*,
> Springer-Verlag.
>
> Sims, C. A., and Zha, T. (1998). Bayesian methods for dynamic mul-
> tivariate models. *International Economic Review*, 39(4):949-968.

SEE ALSO

> `cpredict, forecast.object , predict ,predict.BVAR .`

EXAMPLE

> ```
> pol.mod = VAR(cbind(CP,M2,GDP,CPI)~ar(13), data = log(policy.dat))
> pol.forecast = predict(pol.mod, 12)
> ```

predict.VECM   Use `predict()` on a VECM Object

DESCRIPTION

Performs prediction from a fitted VECM imposing the long run restrictions.

```
predict.VECM(object, n.predict=1, newdata=NULL, seed=100,
             method="asymptotic", n.sim=1000)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"VECM"`.

OPTIONAL ARGUMENTS

`n.predict` : an integer specifying the number of periods to predict ahead.

`newdata` : a data frame containing the future values of exogenous variables, if any, at which predictions are required. This should not include the constant or the time trend term.

`method` : a character string specifying the prediction method. Valid choices are: `"asymptotic"`, for which the forecasts are computed based on asymptotic theory; `"mc"`, for which the forecasts are computed using Monte Carlo simulation; and `"bootstrap"`, for which the forecasts are computed using bootstrap. The default is `"asymptotic"`.

`seed` : an integer between 0 and 1023, or a previous value of `.Random.seed`, which represents the random seed to be used.

`n.sim` : an integer specifying the number of simulations.

VALUE

an object of class `"forecast"`. The `draws` element of the object contains all the simulated draws. See `forecast.object` for details. Sets the value of the `.Random.seed` object in the working directory if `method` is not `"asymptotic"` .

DETAILS

This function is a method for the generic function `predict` for class `"VECM"`. It can be invoked by calling `predict` for an object of the appropriate class, or directly by calling `predict.VECM` regardless of the class of the object.

REFERENCES

Johansen, S. (1995). *Likelihood-based Inference in Cointegrated Vector Autoregressive Models*. Oxford University Press.

SEE ALSO

cpredict.VECM, forecast.object , predict .

EXAMPLE

```
mk.coint = coint(mk.zero2[,1:5], lags=1, trend="rc")
mk.vecm = VECM(mk.coint, coint.rank=3)
predict(mk.vecm, 10)
```

print.archTest   Use `print()` on an `archTest` Object

> `print.archTest(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"archTest"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x`.

print.arima.rob Use `print()` on an `arima.rob` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"arima.rob"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x .

```
print.arima.rob(x, ...)
```

SEE ALSO

`print`.

print.autocorTest Use `print()` on an `autocorTest` object

        `print.autocorTest(x, digits=4, ...)`

This is a method for the function `print()` for objects inheriting from class `"autocorTest"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x`.

print.coint   Use `print()` on a `coint` object

> `print.coint(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"coint"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

print.compare.mgarch Print GARCH Model Comparison

DESCRIPTION

Prints output from a GARCH models comparison. Typically this function is called through the generic function `print` .

```
print.compare.mgarch(x, digits=4, ...)
```

REQUIRED ARGUMENTS

`x` : an object that inherits from class `"compare.mgarch"`. Typically this is the result of a call to the function `compare.mgarch`.

OPTIONAL ARGUMENTS

`digits` : the number of digits for numbers in the output. The default is set to 4.

`...`: any other arguments will be passed to `print` when it is used to print components of the table.

VALUE

returns the input `x`. Prints a table of the model statistics (AIC, BIC and Likelihood) for each model being compared.

SEE ALSO

`compare.mgarch`.

print.FARIMA Use `print()` on a FARIMA Object

> `print.FARIMA(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"FARIMA"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

print.fgarch.model Print FGARCH Model List

DESCRIPTION

Prints the mean and variance equations. Typically, the function is invoked on an object of class `"fgarch.model"` .

`print.fgarch.model(x, digits=4, ...)`

REQUIRED ARGUMENTS

`x`: an object of class `"fgarch.model"`.

OPTIONAL ARGUMENTS

`digits`: number of decimal digits after the decimal point. The default is set to 4.

VALUE

a model list containing the following components:

`formula.mean`: a formula specifying the conditional mean.

`formula.var`: a formula specifying the conditional variance.

SEE ALSO

`print.garch.model`, `print.mgarch.model` .

print.forecast   Use `print()` on a `forecast` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"forecast"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

```
print.forecast(x, digits=4, ...)
```

print.garch.model Print GARCH Model List

DESCRIPTION

Prints the mean and variance equations. Typically, the function is invoked on an object of class `"garch.model"` .

```
print.garch.model(x, digits=4, ...)
```

REQUIRED ARGUMENTS

`x` : an object of class `"garch.model"`.

OPTIONAL ARGUMENTS

`digits` : number of decimal digits after the decimal point. The default is set to 4.

VALUE

a model list containing the following components

`formula.mean`: a formula specifying the conditional mean.

`formula.var`: a formula specifying the conditional variance.

SEE ALSO

`print.mgarch.model`.

EXAMPLE

```
garch11.mod = garch.model(~arma(1,1), ~garch(1,1))
print(garch11.mod)
```

print.garch Print Result of GARCH Model Fitting

DESCRIPTION

This is a method for the function `print` for objects inheriting from class `garch`, which prints a brief description of a fitted object.

```
print.garch(x, digits=4, ...)
```

REQUIRED ARGUMENTS

x : an object of class `"garch"`.

OPTIONAL ARGUMENTS

digits : number of decimal digits after the decimal point. The default is 4.

VALUE

x : with the invisible flag set to prevent from reprinting. prints a summary of the GARCH model.

SEE ALSO

`print.garch.model`, `print.mgarch` .

EXAMPLE

```
hp.fit = garch(hp.s~1, ~garch(1,1))
print(hp.fit)
```

print.gphTest Use `print()` on a `gphTest` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"gphTest"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

```
print.gphTest(x, digits=4, ...)
```

print.heteroTest   Use print() on a heteroTest Object

DESCRIPTION

This is a method for the function print() for objects inheriting from class "heteroTest". See print or print.default for the general behavior of this function and for the interpretation of x.

```
print.heteroTest(x, digits=4, ...)
```

print.impDecomp Use `print()` on an `impDecomp` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"impDecomp"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x`.

```
print.impDecomp(x, digits=4, ...)
```

print.KalmanFil Use `print()` on a KalmanFil Object

> `print.KalmanFil(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class "KalmanFil". See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

print.KalmanSmo Use `print()` on a KalmanSmo Object

> `print.KalmanSmo(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"KalmanSmo"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x`.

print.mfactor   Use `print()` on an `mfactor` Object

DESCRIPTION

>This is a method for the function `print()` for objects inheriting from
>class `"mfactor"`. See `print` or `print.default` for the general behav-
>ior of this function and for the interpretation of x.

>`print.mfactor(x, digits=4, ...)`

print.mgarch.model Print GARCH Model List

DESCRIPTION

Prints the mean and variance equations. Typically, the function is invoked on an object of class `"mgarch.model"` .

```
print.mgarch.model(x, digits=4, ...)
```

REQUIRED ARGUMENTS

x : an object of class `"mgarch.model"`.

OPTIONAL ARGUMENTS

digits : number of decimal digits after the decimal point. The default is set to 4.

The model list specified by `formula.mean` and `formula.var`.

SEE ALSO

`print.garch.model`.

EXAMPLE

```
bekk11.mod = mgarch.model(~arma(1,1), ~bekk(1,1), y.dim=2)
print(bekk11.mod)
```

print.mgarch Print Result of Multivariate GARCH Model Fitting

DESCRIPTION

This is a method for the function `print` for an object inheriting from class `"mgarch"`.

```
print.mgarch(x, digits=4, ...)
```

REQUIRED ARGUMENTS

x : an object of class `"mgarch"`.

OPTIONAL ARGUMENTS

digits : number of decimal digits after the decimal point. The default is set to 4.

VALUE

x : with the invisible flag set to prevent from reprinting. The output of this function includes the formula used to specify the conditional mean equation and the conditional variance equation, the conditional distribution used to fit the model, and the estimated coefficients.

SEE ALSO

`print.garch`.

EXAMPLE

```
hp.ibm = seriesMerge(hp.s, ibm.s)
hp.ibm.mod = mgarch(hp.ibm~1, ~dvec(1,1))
print(hp.ibm.mod)
```

print.mOLS Use `print()` on an `mOLS` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"mOLS"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

```
print.mOLS(x, digits=4, ...)
```

print.NLSUR Use `print()` on an NLSUR Object

> `print.NLSUR(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"NLSUR"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

print.normalTest Use `print()` on a `normalTest` Object

DESCRIPTION
This is a method for the function `print()` for objects inheriting from class `"normalTest"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

```
print.normalTest(x, digits=4, ...)
```

print.OLS Use `print()` on an `OLS` object

```
print.OLS(x, digits=4, ...)
```

This is a method for the function `print()` for objects inheriting from class `"OLS"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x`.

print.outliers Use `print()` on an `outliers` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"outliers"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x .

```
print.outliers(x, ...)
```

SEE ALSO

`outliers`,`print` .

print.RLS Use `print()` on an `RLS` object

> `print.RLS(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"RLS"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x`.

print.rollOLS Use `print()` on a `rollOLS` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"rollOLS"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

```
print.rollOLS(x, digits=4, ...)
```

print.rosTest Use `print()` on a `rosTest` Object

> `print.rosTest(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"rosTest"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

print.SEMIFAR   Use `print()` on a `SEMIFAR` object

> `print.SEMIFAR(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"SEMIFAR"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x`.

print.stationaryTest Use `print()` on a `stationaryTest` Object

> `print.stationaryTest(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"stationaryTest"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x`.

print.summary.arima.rob Use `print()` on a `summary.arima.rob` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"summary.arima.rob"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x` .

```
print.summary.arima.rob(x,  ...)
```

SEE ALSO

`printsummary.arima.rob`.

print.summary.outliers Use `print()` on a `summary.outliers` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"summary.outliers"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x` .

```
print.summary.outliers(x, ...)
```

SEE ALSO

`print`,`summary.outliers` .

print.summaryStats Use `print()` on a summaryStats Object

`print.summaryStats(x, digits=4, ...)`

This is a method for the function `print()` for objects inheriting from class `"summaryStats"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x`.

print.SUR Use `print()` on an SUR Object

> `print.SUR(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"SUR"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of `x`.

print.term.struct Use print() on a term.struct Object

DESCRIPTION

This is a method for the function print() for objects inheriting from class "term.struct". See print or print.default for the general behavior of this function and for the interpretation of x.

```
print.term.struct(x, digits=4, ...)
```

print.unitroot Use `print()` on a `unitroot` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"unitroot"` . See `print` or `print.default` for the general behavior of this function and for the interpretation of x .

```
print.unitroot(x, statistic=NULL, asymptotic=NULL,
               digits=4, ...)
```

REQUIRED ARGUMENTS

x: an object of class `"unitroot"`, usually returned by the function `unitroot`.

OPTIONAL ARGUMENTS

statistic: a character string describing the type of test statistic. Valid choices are `"t"` for t-statistic, and `"n"` for normalized statistic (sometimes referred to as the rho-statistic). By default, the value of `statistic` from fitting x is used.

asymptotic: a logical flag: if `TRUE`, the p-value of the test will be computed based on asymptotic results. By default, the value of `asymptotic` from fitting x is used.

print.VAR Use `print()` on a `VAR` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"VAR"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

```
print.VAR(x, digits=4, ...)
```

print.VECM Use `print()` on a VECM Object

> `print.VECM(x, digits=4, ...)`
>
> This is a method for the function `print()` for objects inheriting from class `"VECM"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

print.waldTest   Use `print()` on a `waldTest` Object

DESCRIPTION

This is a method for the function `print()` for objects inheriting from class `"waldTest"`. See `print` or `print.default` for the general behavior of this function and for the interpretation of x.

```
print.waldTest(x, digits=4, ...)
```

punitroot Cumulative Probability of Unit Root Distributions

DESCRIPTION

Returns cumulative probability of asymptotic or finite sample distribution of unit root test statistics.

```
punitroot(q, n.sample=0, trend="c", statistic="t",
          na.rm=F)
```

REQUIRED ARGUMENTS

`q` : vector of quantiles or test statistics. Missing values (`NA`) are allowed.

OPTIONAL ARGUMENTS

`n.sample` : the number of observations in the sample from which the test statistics are computed. Specify `n.sample=0` for asymptotic cumulative probabilities. The default is 0.

`trend` : a character string describing the regression from which the test statistics are computed. Valid choices are: `"nc"` for a regression with no intercept (constant) nor time trend, and `"c"` for a regression with an intercept (constant) but no time trend, `"ct"` for a regression with an intercept (constant) and a time trend. The default is `"c"`.

`statistic` : a character string describing the type of test statistic. Valid choices are `"t"` for t-statistic, and `"n"` for normalized statistic (sometimes referred to as the rho-statistic). The default is `"t"`.

`na.rm` : a logical flag: if `TRUE`, missing values will be removed before computing the tests. The default is `FALSE`.

VALUE

vector of cumulative probabilities from the specified distribution.

REFERENCES

Dickey, D. A., and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427-431.

MacKinnon, J. G. (1996). Numerical distribution functions for unit root and cointegration tests. *Journal of Applied Econometrics*, 11:601-618.

Phillips, P. C. B., and Perron, P. (1988). Testing for a unit root in time series regression. *Biometrika*, 75(2):335-346.

SEE ALSO

`pcoint`,`qcoint` ,`qunitroot` ,`unitroot` .

EXAMPLE

```
# asymptotic cumulative probability of t-statistic
punitroot(1.2836, trend="nc", statistic="t")

# finite sample cumulative probability of n-statistic
punitroot(1.2836, n.sample=100, trend="nc", statistic="n")
```

qcoint Quantiles of Cointegration Distributions

DESCRIPTION
> Returns quantiles of asymptotic or finite sample distributions of cointegration test statistics, given the probabilities.

```
qcoint(p, n.sample=0, n.series=2, estimator="c",
       statistic="t")
```

REQUIRED ARGUMENTS

> p : vector of probabilities. Missing values (`NA`) are allowed.

OPTIONAL ARGUMENTS

> n.sample : the number of observations in the sample from which the quantiles are to be computed. Specify `n.sample=0` for asymptotic quantiles. The default is 0.

> n.series : the number of series used in the cointegration test. The default is 2.

> estimator : a character string describing the regression from which the quantiles are to be computed. Valid choices are: `"nc"` for a regression with no intercept (constant) nor time trend, and `"c"` for a regression with an intercept (constant) but no time trend, `"ct"` for a regression with an intercept (constant) and a time trend. The default is `"c"`.

> statistic : a character string describing the type of test statistic. Valid choices are `"t"` for t-statistic, and `"n"` for normalized statistic (sometimes referred to as the rho-statistic). The default is `"t"`.

VALUE
> vector of quantiles from the specified distribution.

DETAILS
> The functions `pcoint` and `qcoint` only apply to the cointegration tests computed using the Engle-Granger procedure. They are based on the response surface regression coefficients provided by MacKinnon (1996) and do not apply to Johansen's rank tests.

REFERENCES
> MacKinnon, J. G. (1996). Numerical distribution functions for unit root and cointegration tests. *Journal of Applied Econometrics*, 11:601-618.

SEE ALSO
> coint, pcoint ,punitroot ,qunitroot .

qplot Exploratory QQ-Plot for Extreme Value Analysis

DESCRIPTION

Generates a QQ-plot for threshold data against a reference distribution
(exponential or generalized Pareto distribution).

```
qplot(data, xi=0, trim=NA, threshold=NA, line=T, labels=T,
      ...)
```

REQUIRED ARGUMENTS

data: a numeric vector or a `"timeSeries"` object with a numeric vector in
the data slot.

OPTIONAL ARGUMENTS

xi : a numeric value of `xi` parameter in a generalized Pareto distribution.
The default is `0`.

trim : a numeric value at which data are to be right-truncated. If `NA`, the
data is not truncated. The default is `NA`.

threshold : a numeric value at which data are to be left-truncated. If `NA`, the
data is not truncated. The default is `NA`.

line : a logical flag: if `TRUE`, a simple regression line is added. The default
is `TRUE`.

labels : a logical flag: if `TRUE`, the axes are labeled. The default is `TRUE`.

DETAILS

If `xi` is zero the reference distribution is the exponential distribution;
if `xi` is non-zero the reference distribution is the generalized Pareto
distribution with the parameter `xi`. In the case of exponential dis-
tribution, the plot is interpreted as follows: concave departure from
a straight line is a sign of heavy-tailed behavior, while convex depar-
ture shows thin-tailed behavior. A QQ-plot for threshold data will be
drawn on a graphical device.

SEE ALSO

`meplot`,`gpd` .

EXAMPLE

```
# QQplot of heavy-tailed Danish fire insurance data
qplot(danish)
```

qqPlot Trellis QQ-Plot for Various Distributions

DESCRIPTION

Generates a trellis object representing the qq-plot of selected distributions.

```
qqPlot(x, dof, strip.text="", hgrid=F, vgrid=F, id.n=3,
       distribution="normal", prepanel=prepanel.qqPlot,
       panel=panel.qqPlot, xlab=NULL, ylab=NULL, ...)
```

REQUIRED ARGUMENTS

x : a rectangular numeric object. If x represents a multivariate data, then the variables must be in columns.

dof : the degree of freedom for Student-t distribution. This argument is not required for other distributions.

OPTIONAL ARGUMENTS

strip.text : a character string which will appear in the strip of the trellis plot. The default is "", which draws nothing in the strip. If x has more than one column, then strip.text must be a character vector with length equal to the number of columns of x.

hgrid : a logical flag: if TRUE, horizontal grids will be drawn on the trellis plot. The default is FALSE.

vgrid : a logical flag: if TRUE, vertical grids will be drawn on the trellis plot. The default is FALSE.

id.n : number of points (must be less than the number of observations) to be identified in the qq-plot. These will be the id.n largest points in absolute value. Set to FALSE if no identified points are desired. The default is 3.

distribution : a character string specifying the comparison distribution to use. Current valid choices are: "normal" for normal distribution, "t" for Student-t distribution, and "double.exp" for double exponential distribution. The default is to use normal distribution.

panel : a function of two arguments, x and y, that draws the data display in each panel. See the help file for trellis.args for details.

prepanel : an optional function that is called prior to plotting in order to set up appropriate axis scales and aspect ratios. See the help file for trellis.args for details.

xlab : a character string for label on x-axis.

ylab : a character string for label on y-axis.

... : any other optional arguments that can be passed down to low level plot functions.

VALUE

an object of class `trellis`, which is automatically plotted by `print.trellis`.

DETAILS

If `distribution` is set to `"t"` when calling `qqPlot`, then an optional argument`dof` must be supplied, which is a vector specifying the degree of freedom for each column of `x`. If this function is invoked directly without saving the returned object, a trellis device will be open to draw the plot if one is not available yet.

SEE ALSO

qq,qqplot ,trellis.args .

EXAMPLE

```
# normal QQ-plot of daily Dell stock returns
qqPlot(dell.s, strip="Daily Dell Returns", hgrid=T, vgrid=T)

# student-t QQ-plot of HP and IBM stock returns
qqPlot(seriesMerge(hp.s, ibm.s), strip=c("HP","IBM"), dof=c(6,6), distr="t")
```

quant Calculate and Plot GPD Tail Estimate of a High Quantile

DESCRIPTION

Calculates and generates a plot showing how the estimate of a high quantile in the tail of a data set based on the GPD approximation varies with threshold or number of extremes.

```
quant(data, p=0.99, models=30, start=15, end=500,
    reverse=T, ci=0.95, auto.scale=T, labels=T,
    table=F, ...)
```

REQUIRED ARGUMENTS

  data: a numeric vector or `"timeSeries"` object.

OPTIONAL ARGUMENTS

  p: a number between 0 and 1 specifying the desired probability for quantile estimate (e.g. 0.99 gives 99th percentile). The default is 0.99.

  models: an integer specifying the number of consecutive GPD models to be fitted. The default is 30.

  start: an integer specifying the lowest number of exceedances in a series of GPD models to fit. The default is 15.

  end: an integer specifying the maximum number of exceedances in a series of GPD models to fit. The default is 500.

  reverse: a logical flag: if `TRUE` plot is to be by increasing threshold; if `FALSE` by increasing number of extremes. The default is `TRUE`.

  ci: a number between 0 and 1 specifying the probability for asymptotic confidence band; for no confidence band set `ci` to `FALSE`. The default is 0.95.

  auto.scale: a logical flag: if `TRUE`, plot is automatically scaled; if `FALSE`, `xlim` and `ylim` graphical parameters may be entered. The default is `TRUE`.

  labels: a logical flag: if `TRUE`, axes are labeled. Labels are ignored if `FALSE`. The default is `TRUE`.

  table: a logical flag: if `TRUE`, a table of results is printed. The default is `FALSE`.

  ... : any optional arguments that can be passed down to the `plot` function.

VALUE

a matrix which contains the following columns:

`threshold` : the thresholds.

`qest` : the high quantile estimates.

`exceedances` : the number of extremes.

`lower` : the lower bounds of confidence interval.

`upper` : the upper bounds of confidence interval. A plot of high quantile estimates and their confidence intervals based on estimated GPD models will be drawn on a graphical device over increasing thresholds or number of extremes.

DETAILS

For every model the `gpd` function is called, and thus evaluation may be slow. Confidence intervals are constructed by the Wald method (which is the fastest).

SEE ALSO

gpd, `plot.gpd` , `gpd.q` , `shape` .

EXAMPLE

```
# Estimates of the 99.9th percentile of the Danish losses using
# the GPD model with various thresholds
quant(danish, 0.999)
```

qunitroot   Quantiles of Unit Root Distributions

DESCRIPTION

Returns quantiles of asymptotic or finite sample distributions of unit root test statistics, given the probabilities.

```
qunitroot(p, n.sample=0, trend="c", statistic="t",
          na.rm=F)
```

REQUIRED ARGUMENTS

`p` : vector of probabilities. Missing values (`NA`) are allowed.

OPTIONAL ARGUMENTS

`n.sample` : the number of observations in the sample from which the quantiles are to be computed. Specify `n.sample=0` for asymptotic quantiles. The default is 0.

`trend` : a character string describing the regression from which the quantiles are to be computed. Valid choices are: `"nc"` for a regression with no intercept (constant) nor time trend, and `"c"` for a regression with an intercept (constant) but no time trend, `"ct"` for a regression with an intercept (constant) and a time trend. The default is `"c"`.

`statistic` : a character string describing the type of test statistic. Valid choices are `"t"` for t-statistic, and `"n"` for normalized statistic (sometimes referred to as the rho-statistic). The default is `"t"`.

`na.rm` : a logical flag: if `TRUE`, missing values will be removed before computing the tests. The default is `FALSE`.

VALUE

vector of quantiles from the specified distribution.

REFERENCES

Dickey, D. A., and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427-431.

MacKinnon, J. G. (1996). Numerical distribution functions for unit root and cointegration tests. *Journal of Applied Econometrics*, 11:601-618.

Phillips, P. C. B., and Perron, P. (1988). Testing for a unit root in time series regression. *Biometrika*, 75(2):335-346.

SEE ALSO

`pcoint`,`qcoint` ,`punitroot` ,`unitroot` .

EXAMPLE

```
# asymptotic quantile of t-statistic
qunitroot(0.95, trend="nc", statistic="t")

# finite sample quantile of n-statistic
qunitroot(0.95, n.sample=100, trend="nc", statistic="n")
```

rafPlot Trellis Plot of Response and Fitted Values

DESCRIPTION

Generates a trellis object representing two time series, for example, response and fitted values from a model object.

```
rafPlot(response, fits, strip.text="", hgrid=F, vgrid=F,
        prepanel=prepanel.rafPlot, panel=panel.rafPlot,
        main="", ...)
```

REQUIRED ARGUMENTS

response : a numeric vector or a `"timeSeries"` object with a numeric vector in the data slot. This can be the response from a fitted model object.

fits : a numeric vector or a `"timeSeries"` object with a numeric vector in the data slot. This can be the fitted values from a fitted model object.

OPTIONAL ARGUMENTS

strip.text : a character string which will appear in the strip of the trellis plot. The default is `""`, which draws nothing in the strip.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the trellis plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the trellis plot. The default is `FALSE`.

panel : a function of two arguments, `x` and `y`, that draws the data display in each panel. See the help file for `trellis.args` for details.

prepanel : an optional function that is called prior to plotting in order to set up appropriate axis scales and aspect ratios. See the help file for `trellis.args` for details.

main : a character string giving the main title to be drawn on the plot. The default is `""`, which draws nothing.

... : any other optional arguments that can be passed down to low level plot functions.

VALUE

an object of class `trellis`, which is automatically plotted by `print.trellis`.

DETAILS

This function is called by many `plot` methods to produce the Response and Fitted Values plot. However, it can also be used to compare any two time series of equal length. If this function is invoked directly

without saving the returned object, a trellis device will be open to draw the plot if one is not available yet.

SEE ALSO

rvfPlot,trellis.args ,trellisPlot.timeSeries .

EXAMPLE

```
# Comparison of high and low prices.
rafPlot(msft.dat[,"High"], msft.dat[,"Low"], strip="High & Low")
```

records   Calculate Record Development

DESCRIPTION

Creates a data frame showing the development of records in a data set and calculates the expected behavior for iid data.

```
records(data, plot=T, conf.level=0.95)
```

REQUIRED ARGUMENTS

data : a numeric vector, or a `"timeSeries"` object with a numeric vector in the data slot.

OPTIONAL ARGUMENTS

plot : a logical flag: if `TRUE`, a plot of record development is created. The default is `TRUE`.

conf.level : a number between 0 and 1 specifying the confidence level for record development plot. The default is `0.95`.

VALUE

a data frame, or a `"timeSeries"` data frame, with the following components:

number : the index of records.

record : the record values.

trial : the record times.

expected : the corresponding expected values of records.

se : the standard deviations of records. Both the fourth and fifth columns are computed for iid data.

DETAILS

The `records` function counts the records and records the observations at which they occur. The records are compared with the expected behavior for iid data. If `plot=T`, the record development plot will be drawn on a graphical device.

REFERENCES

Embrechts, P., Kluppelberg, C., and Mikosch, T. (1997). *Modelling Extremal Events for Insurance and Finance*. Springer-Verlag.

EXAMPLE

```
# Record fire insurance losses in Denmark
records(danish)
```

residuals.FARIMA Use `residuals()` on a `FARIMA` Object

```
residuals.FARIMA(object, ...)
```

This is a method for the function `residuals()` for objects inheriting from class `"FARIMA"`. See `residuals` or `residuals.default` for the general behavior of this function and for the interpretation of `object`.

residuals.mfactor   Use `residuals()` on an `mfactor` Object

DESCRIPTION

This is a method for the function `residuals()` for objects inheriting
from class `"mfactor"`. See `residuals` or `residuals.default` for the
general behavior of this function and for the interpretation of `object`.

```
residuals.mfactor(object, ...)
```

VALUE

the residuals matrix of the multi-factor model.

SEE ALSO

`residuals`.

residuals.mgarch Extracts Residuals From a Fitted GARCH Model

DESCRIPTION

This is a method for function `residuals` for an object inheriting from class `mgarch` or `garch`.

```
residuals(x, standardize=T)
```

REQUIRED ARGUMENTS

`x` : an object of class `"garch"` or `"mgarch"`.

OPTIONAL ARGUMENTS

`standardize` : a logical flag: if `TRUE`, the residuals will be divided by the square root of the estimated conditional variance sequence.

VALUE

a numerical sequence of residuals from fitting the GARCH model, the same length as that of the data.

SEE ALSO

`residuals`.

EXAMPLE

```
hp.ibm = seriesMerge(hp.s, ibm.s)
hp.ibm.mod = mgarch(hp.ibm~1, ~dvec(1,1))
resid(hp.ibm.mod)
```

residuals.OLS   Use `residuals()` on an `OLS` Object

> `residuals.OLS(object, ...)`
>
> This is a method for the function `residuals()` for objects inheriting from class `"OLS"`. See `residuals` or `residuals.default` for the general behavior of this function and for the interpretation of `object`.

residuals.RLS Use `residuals()` on an RLS Object

DESCRIPTION

This is a method for the function `residuals()` for objects inheriting from class `"RLS"`. See `residuals` or `residuals.default` for the general behavior of this function and for the interpretation of `object`.

```
residuals.RLS(object, ...)
```

VALUE

the recursive residuals.

SEE ALSO

`residuals`.

residuals.SEMIFAR Use `residuals()` on a SEMIFAR Object

`residuals.SEMIFAR(object, ...)`

This is a method for the function `residuals()` for objects inheriting from class `"SEMIFAR"`. See `residuals` or `residuals.default` for the general behavior of this function and for the interpretation of `object`.

residuals.SUR Use `residuals()` on a SUR Object

DESCRIPTION

This is a method for the function `residuals()` for objects inheriting from class `"SUR"`. See `residuals` or `residuals.default` for the general behavior of this function and for the interpretation of `object`.

```
residuals.SUR(object, ...)
```

VALUE

the residuals matrix of the SUR model.

SEE ALSO

`residuals`.

residuals.unitroot Use `residuals()` on a `unitroot` Object

> `residuals.unitroot(object, ...)`
>
> This is a method for the function `residuals()` for objects inheriting from class `"unitroot"`. See `residuals` or `residuals.default` for the general behavior of this function and for the interpretation of `object`.

residuals.VAR Use `residuals()` on a `VAR` Object

DESCRIPTION

This is a method for the function `residuals()` for objects inheriting from class `"VAR"`. See `residuals` or `residuals.default` for the general behavior of this function and for the interpretation of `object`.

```
residuals.VAR(object, ...)
```

VALUE

the residuals matrix of the VAR model.

SEE ALSO

`residuals`.

rf.30day   Nominal Interest Rate Data

SUMMARY

This is a monthly `"timeSeries"` object from July 1926 to December
2000, representing nominal interest rate on 30-day U.S. Treasury bills.

riskmeasures Calculate Quantiles and Expected Shortfalls

DESCRIPTION

Calculates the point estimates of prescribed quantiles and expected shortfalls using a `"gpd"` object as returned by the `gpd` function.

```
riskmeasures(out, p)
```

REQUIRED ARGUMENTS

`out:` a `"gpd"` object as returned by the `gpd` function.

`p:` a numeric vector specifying the probability levels.

VALUE

a matrix containing the following three columns: probability level, quantile estimate, and shortfall estimate.

DETAILS

This function simply calculates point estimates without the confidence intervals for the risk measures. If confidence levels are required, use `gpd.q` and `gpd.sfall` which are much slower.

SEE ALSO

`gpd`, `plot.gpd` , `tailplot` , `gpd.q` , `gpd.sfall` .

EXAMPLE

```
## gives estimates of 0.999 and 0.9999 quantiles of Danish
## loss distribution as well as the associated expected
## shortfall estimates
out = gpd(danish, 10)
riskmeasures(out, c(0.999, 0.9999))
```

rlevel.gev Return Level and Confidence Interval for GEV Model

DESCRIPTION

Calculates the k-block return level and its confidence interval based on a GEV model for block maxima, where k is specified by the user. The k-block return level is the level exceeded once every k blocks on average.

```
rlevel.gev(out, k.blocks=20, type="profile", max.fcal=100,
           max.iter=100)
```

REQUIRED ARGUMENTS

out: an object of class "gev". This is usually returned by the gev function.

OPTIONAL ARGUMENTS

k.blocks: an integer specifying a particular return level to be estimated. The default is 20.

type: a character string specifying which of thw two types of plots to be generated. It takes the value of either "profile" or "RetLevel". The default is "profile".

max.fcal : an integer specifying the maximum number of function evaluations allowed in maximum likelihood estimation. The default is 100.

max.iter : an integer specifying the maximum number of iterations allowed in maximum likelihood estimation. The default is 100.

VALUE

a list which contains the following components depending on the argument type. If type="profile", it contains the following components:

Range : a data frame which contains a return level vector (and a corresponding ML value vector) over a 99.99% confidence interval range.

rlevel : the point estimate of return level.

If type="RetLevel", it contains the following components:

LowerCB : lower 95% bound of confidence interval of the return level.

rlevel : the point estimate of return level.

LowerCB : upper 95% bound of confidence interval of the return level.

DETAILS

The GEV likelihood is reparameterized in terms of the unknown return level and profile likelihood arguments are used to construct a confidence interval. If `type="profile"`, a time series plot of block maxima data with the return level and its 95% confidence interval will be drawn on a graphical device. If `type="RetLevel"`, a graph of the profile likelihood curve with the return level is drawn on a graphical device.

SEE ALSO

`gev,plot.gev` .

EXAMPLE

```
# Fit GEV to monthly maxima of daily returns on BMW share price
out = gev(bmw, "month")
# Calculate the 40 month return level
rlevel.gev(out, 40)
```

RLS.fit Fitting of a Recursive Least Squares Model

DESCRIPTION

Returns necessary components for an "OLS" object representing the recursive least squares fit. It is called by the function OLS, and its direct use is not recommended.

```
RLS.fit(x, y)
```

REQUIRED ARGUMENTS

x : a numeric vector or matrix for the predictors in the recursive least squares estimation. Typically, but not necessarily, x will be the model matrix generated by RLS function.

y : a numeric vector or matrix for the response in the recursive least squares estimation.

SEE ALSO

lm.fit.qr, RLS , RLS.object .

RLS.object Recursive Least Squares Model Object

DESCRIPTION

These are objects of class `"RLS"`. They represent the recursive fit of an ordinary least squares model.

GENERATION

This class of objects is returned from the `RLS` function.

METHODS

The `"RLS"` class of objects currently has methods for the following generic functions:

`coef`, `residuals`, `plot`, `print`.

STRUCTURE

The following components must be present in a legitimate `"RLS"` object:

`call` : an image of the call that produced the object, but with the arguments all named and with the actual formula included as the `formula` argument.

`coef` : a matrix giving the recursive coefficients, with each row being the coefficients from each fit.

`residuals` : the recursive residuals. This is used to construct the CUSUM tests.

`terms` : an object of mode `expression` and class `term` summarizing the formula. Used by various methods, but typically not of direct relevance to users.

`positions` : the row indices of the original `data` argument, or the positions of `data` if it is a `"timeSeries"` object.

`qr` : the last QR decomposition in the recursive fit.

`k` : the number of estimated coefficients for each model fit.

SEE ALSO

`lm`, `lm.object` , `OLS` ,`OLS.object` ,`RLS` .

RLS Recursive Least Squares Estimation

DESCRIPTION
Estimates an ordinary least squares model recursively.

```
RLS(formula, data, subset, na.rm=F, contrasts=NULL, start=NULL,
    end=NULL, ...)
```

REQUIRED ARGUMENTS

formula : a `formula` object, with the response on the left of a $\sim$ operator, and the terms, separated by + operators, on the right. The variables cannot be `"timeSeries"` objects if `data` argument is not specified.

OPTIONAL ARGUMENTS

data : a data frame or `"timeSeries"` data frame in which to interpret the variables named in the `formula` and `subset` arguments. If `data` is missing, the variables in the model formula should be in the search path, and cannot be `"timeSeries"` objects.

subset : an expression specifying which subset of observations should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), a numeric vector indicating the observation numbers to be included, or a character vector of the observation names that should be included. All observations are included by default.

na.rm : a logical flag: if `TRUE`, missing values will be removed before fitting the model. The default is `FALSE`.

contrasts : a list giving contrasts for some or all of the factors appearing in the model formula. An element in the list should have the same name as the factor variable it encodes, and it should be either a contrast matrix (any full-rank matrix with as many rows as there are levels in the factor), or a function that computes such a matrix given the number of levels.

start : a character string which can be passed to `timeDate` function to specify the starting date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

end : a character string which can be passed to `timeDate` function to specify the ending date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

... : any optional arguments that can be passed to the `timeDate` function to parse the `start` or `end` specification.

VALUE

an object of class "RLS" representing the recursive LS fit. See `RLS.object` for details.

DETAILS

Recursive least squares estimation is usually performed to diagnose structural change in the linear model. For this purpose, `cusumTest` function can be applied on a "RLS" object to perform CUSUM test.

REFERENCES

Brown, R. L., Durbin, J., and Evans, J. M. (1975). Techniques for testing the constancy of regression relationships over time. *Journal of the Royal Statistical Society B*, 37:141-192.

SEE ALSO

`cusumTest,lm` , `OLS` ,`RLS.object` , `plot.RLS` , `rollOLS` .

EXAMPLE

```
tmp.dat = data.frame(Loss=stack.loss, stack.x)
RLS(Loss~Air.Flow+Water.Temp+Acid.Conc., data=tmp.dat)
```

rollOLS.fit Fitting of Rolling OLS Models

DESCRIPTION

Called by `rollOLS` to fit the models. It is not supposed to be called by the users directly.

```
rollOLS.fit(x, y, width=NULL, incr=1, tau=1e-10, trace=T)
```

REQUIRED ARGUMENTS

X, Y : numeric matrices representing the regressors and multivariate response respectively.

OPTIONAL ARGUMENTS

width : an integer specifying the width of each rolling window. If `width` is NULL, it is set to `max(p+1, floor(n/5))`.

incr : an integer specifying the number of observations to move ahead for each rolling window. The default is 1.

tau : a small number which represents the singularity tolerance. The default is `1e-10`.

trace : a logical flag: if `TRUE`, a message will be printed on the screen when each rolling sample is estimated. The default is `TRUE`.

rollOLS.object Rolling Ordinary Least Squares Model Object

DESCRIPTION

These are objects of class `"rollOLS"`. They represent the fit of a rolling ordinary least squares estimation.

GENERATION

This class of objects is returned from the `rollOLS` function.

METHODS

The `"rollOLS"` class of objects currently has methods for the following generic functions:

`coef`, `plot`, `predict`, `print`, `summary`.

STRUCTURE

The following components must be present in a legitimate `"rollOLS"` object:

call : an image of the call that produced the object, but with the arguments all named and with the actual formula included as the `formula` argument.

coef : the coefficient matrix of the rolling regression, with each row being the coefficient estimates from each rolling window.

stddev : the standard errors of `coef`, with each row from each rolling window.

sigma : a vector giving the residual scale estimate from each rolling window.

rdf : the number of degrees of freedom for residuals in each rolling window.

terms : an object of mode `expression` and class `term` summarizing the formula. Used by various methods, but typically not of direct relevance to users.

contrasts : a list containing sufficient information to construct the contrasts used to fit any factors occurring in the model. The list contains entries that are either matrices or character vectors. When a factor is coded by contrasts, the corresponding contrast matrix is stored in this list. Factors that appear only as dummy variables and variables in the model that are matrices correspond to character vectors in the list. The character vector has the level names for a factor or the column labels for a matrix.

positions : a character vector giving the starting and ending positions.

width : an integer giving the width of each rolling window.

incr : an integer giving the number of observations for which each window is moved ahead.

nwin : an integer giving the total number of windows used.

SEE ALSO

OLS, RLS ,rollOLS .

rollOLS Rolling Ordinary Least Squares Estimation

DESCRIPTION

   Performs rolling ordinary least squares estimation.

```
rollOLS(formula, data, subset, na.rm=F, contrasts=NULL,
        start=NULL, end=NULL, width=NULL, incr=1, tau=1e-10,
        trace=T, ...)
```

REQUIRED ARGUMENTS

formula : a `formula` object, with the response on the left of a $\sim$ operator, and the terms, separated by + operators, on the right. The variables cannot be `"timeSeries"` objects if `data` argument is not specified.

data : a data frame or `"timeSeries"` data frame in which to interpret the variables named in the `formula` and `subset` arguments. If `data` is missing, the variables in the model formula should be in the search path, and cannot be `"timeSeries"` objects.

OPTIONAL ARGUMENTS

subset : an expression specifying which subset of observations should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), a numeric vector indicating the observation numbers to be included, or a character vector of the observation names that should be included. All observations are included by default.

na.rm : a logical flag: if `TRUE`, missing values will be removed before fitting the model. The default is `FALSE`.

contrasts : a list giving contrasts for some or all of the factors appearing in the model formula. An element in the list should have the same name as the factor variable it encodes, and it should be either a contrast matrix (any full-rank matrix with as many rows as there are levels in the factor), or a function that computes such a matrix given the number of levels.

start : a character string which can be passed to `timeDate` function to specify the starting date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

end : a character string which can be passed to `timeDate` function to specify the ending date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

width : an integer specifying the width of each rolling window. If `width` is NULL, it is set to `max(p+1, floor(n/5))`.

incr : an integer specifying the number of observations to move ahead for each rolling window. The default is 1.

tau : a small number which represents the singularity tolerance. The default is `1e-10`.

trace : a logical flag: if TRUE, a message will be printed on the screen when each rolling sample is estimated. The default is TRUE.

... : any optional arguments that can be passed to the `timeDate` function to parse the `start` or `end` specification.

VALUE

an object of class `"rollOLS"` representing the fit. See `rollOLS.object` for details.

DETAILS

An efficient block addition and deletion algorithm is used for fast execution.

REFERENCES

Chambers, J. M. (1975). Updating methods for linear models for the addition or deletion of observations. *Survey of Statistical Design and Linear Models*, J. N. Srivastava (ed.), North-Holland Publishing Company.

SEE ALSO

OLS,RLS ,rollOLS.object .

EXAMPLE

```
stack.dat = data.frame(Loss=stack.loss, stack.x)
rollOLS(Loss~Water.Temp, data=stack.dat, width=6, incr=2)
```

roll Generic Rolling Estimation

DESCRIPTION

Performs the rolling estimation based on a user-given function.

```
roll(FUN, data, width, incr=1, start=NULL, end=NULL,
     na.rm=F, save.list=NULL, arg.data="data", trace=T, ...)
```

REQUIRED ARGUMENTS

FUN : a character string giving the name of the S function for which the rolling estimation will be performed. This function must take an optional argument called `data`.

data : a data frame or `"timeSeries"` data frame upon which the rolling estimation will be applied. Each time we will take a rolling sample from `data`, and apply the function FUN on the rolling sample.

width : an integer specifying the width of each rolling window.

OPTIONAL ARGUMENTS

incr : an integer specifying the number of observations to move ahead for each rolling window. The default is 1.

start : a character string which can be passed to `timeDate` function to specify the starting date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

end : a character string which can be passed to `timeDate` function to specify the ending date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

na.rm : a logical flag: if `TRUE`, missing values will be removed before running the rolling estimation. The default is `FALSE`.

save.list : a character vector giving the names of the components of the returned value of FUN which will be saved. If `NULL`, everything in the returned value will be saved.

arg.data : a character string giving the name used for the data argument by FUN. For example, if FUN is `"mean"`, then `arg.data` should be set to `"x"`. The default is `"data"`.

trace : a logical flag: if `TRUE`, a message will be printed on the screen when each rolling sample is estimated. The default is `TRUE`.

... : any other optional argument that need to be passed to FUN.

VALUE

> an object of class `"roll"`. This will contain all those components of the returned value of `FUN` whose names correspond to those in `save.list`, and the following components:

`call`: an image of the original function call.

`pos`: a character vector with length 2, giving the starting and ending position of the data used.

SEE ALSO

> `rollOLS`.

EXAMPLE

```
# a brute force rolling estimation
stack.dat = data.frame(Loss=stack.loss, stack.x)
roll("OLS", stack.dat, 7, formula=Loss~Air.Flow+Water.Temp,
     save.list="coef")
```

rollVar Rolling/Moving Sample Statistics

DESCRIPTION

Returns rolling or moving sample maximum, sample minimum, or sample variance.

```
rollMax(x, n=9, trim=T, na.rm=F)
rollMin(x, n=9, trim=T, na.rm=F)
rollVar(x, n=9, trim=T, unbiased=T, na.rm=F)
```

REQUIRED ARGUMENTS

x : a numeric vector, matrix, or a `"timeSeries"` object with a numeric object in the data slot.

OPTIONAL ARGUMENTS

n : an integer specifying the number of periods or terms to use in each rolling sample. The default uses a 9-term rolling/moving sample.

trim : a logical flag: if `TRUE`, the first `n-1` missing values in the returned object will be removed; if `FALSE`, they will be saved in the returned object. The default is `TRUE`.

unbiased : a logical flag: if `TRUE`, the unbiased sample variance is returned. The default is `TRUE`.

na.rm : a logical flag: if `TRUE`, missing values in `x` will be removed before computation. The default is `FALSE`.

VALUE

a numeric vector, matrix, or a `"timeSeries"` object with a numeric vector or matrix in the data slot. `rollMax` returns the rolling sample maximum, `rollMin` returns rolling sample minimum, and `rollVar` returns rolling sample variance. If there are more than one column in `x` then the column-wise results are returned.

SEE ALSO

EWMA,SMA ,cummax ,iMVar .

EXAMPLE

```
rollVar(1:100, 11, unbiased=F)
```

rosTest R/S Test for Long Memory or Long-Range Dependence

DESCRIPTION

Returns the (modified) R/S test for long memory or long range dependence.

```
rosTest(x, bandwidth=NULL, window="bartlett", na.rm=F)
```

REQUIRED ARGUMENTS

x : a vector, matrix, data frame, or a `"timeSeries"` object with a numeric object in the data slot, representing either a univariate or a multivariate time series.

OPTIONAL ARGUMENTS

bandwidth : an integer that specifies the bandwith to be used. If `bandwidth` is zero, then traditional R/S statistics are returned; if `bandwidth` is positive, then Lo's modified R/S statistics are returned. The default is set to `4*(n/100)^0.25`, where `n` is the sample size of `x`.

window : a character string that specifies the type of window to be used. Currently the only valid choices are `"Bartlett"` for Bartlett's triangular window, or `"rectangular"` for a rectangular window. This is ignored if `bandwidth=0`. The default is `"Bartlett"`.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computing the test statistics. The default is `FALSE`.

VALUE

an object of class `"rosTest"`, which contains the following components:

stat : a numeric vector giving the R/S statistics.

bandwidth : an integer giving the chosen bandwidth.

n : an integer giving the sample size of `x`.

na : an integer giving the number of missing values in `x`.

DETAILS

If the input `x` is multivariate, then the R/S test statistics are all the series simultaneously.

REFERENCES

Lo, A. W. (1991). Long-term memory in stock market prices. *Econometrica*, 59:1279:1313.

SEE ALSO

> `FARIMA`, `d.ros` , `d.pgram` , `d.whittle` , `gphTest` .

EXAMPLE

> ```
> # R/S test on absolute DELL stock returns.
> rosTest(abs(dell.s))
> ```

rvfPlot Trellis Plot of Response versus Fitted Values

DESCRIPTION

Generates a trellis object representing two time series, for example, response and fitted values from a model object.

```
rvfPlot(response, fits, strip.text="", hgrid=F, vgrid=F,
        prepanel=prepanel.rvfPlot, panel=panel.rvfPlot,
        id.n=3, ...)
```

REQUIRED ARGUMENTS

response : a numeric vector or a `"timeSeries"` object with a numeric vector in the data slot. This can be the response variable from a fitted model object.

fits : a numeric vector or a `"timeSeries"` object with a numeric vector in the data slot. This can be the fitted values from a fitted model object.

OPTIONAL ARGUMENTS

strip.text : a character string which will appear in the strip of the trellis plot. The default is `""`, which draws nothing in the strip.

hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the trellis plot. The default is `FALSE`.

vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the trellis plot. The default is `FALSE`.

panel : a function of two arguments, x and y, that draws the data display in each panel. See the help file for `trellis.args` for details.

prepanel : an optional function that is called prior to plotting in order to set up appropriate axis scales and aspect ratios. See the help file for `trellis.args` for details.

id.n : number of points (must be less than the number of observations) to be identified in the plot. These will be the `id.n` largest points in absolute difference of `response` and `fits`. Set to `FALSE` if no identified points are desired. The default is 3.

... : any other optional arguments that can be passed down to low level plot functions.

VALUE

an object of class `trellis`, which is automatically plotted by `print.trellis`.

DETAILS

This function is called by many `plot` methods to produce the Response vs. Fitted Values plot. However, it can also be used to compare any two time series of equal length. If this function is invoked directly without saving the returned object, a trellis device will be open to draw the plot if one is not available yet.

SEE ALSO

rafPlot,trellis.args ,trellisPlot.timeSeries .

EXAMPLE

```
# Comparison of high and low prices.
rvfPlot(msft.dat[,"High"], msft.dat[,"Low"], strip="High vs. Low",
        xlab="Low", ylab="High")
```

sample.LMOM Sample L-Moments

DESCRIPTION

Computes unbiased estimates of sample mean, second, L-moment, L-skewness and L-kurtosis.

```
sample.LMOM(x)
```

REQUIRED ARGUMENTS

x : a numeric vector of data points or `"timeSeries"` object. Missing values (NAs) are not allowed.

VALUE

a vector of length four with the estimates `l1`, `l2`, `tau3`, and `tau4`.

REFERENCES

Hosking, J. R. M. (1986). The theory of probability weighted moments. Research Report RC12210, IBM Research, Yorktown Heights, NY.

Hosking, J. R. M. (1990). L-moments: analysis and estimation of distributions using linear combinations of order statistics. *Journal of Royal Statistical Society*, 52(1):105-124.

SEE ALSO

`PlotPos.LMOM`.

EXAMPLE

```
sample.LMOM(rgev(500, sigma=3.5, mu=0, xi=0.4))
```

screeplot.mfactor   Screeplot of an `mfactor` Object

DESCRIPTION

Generates a screeplot of an `"mfactor"` object.

```
screeplot.mfactor(mf, variables, cumulative=T, style="bar",
                  main="", ...)
```

REQUIRED ARGUMENTS

mf : an object of class `"mfactor"`. This is usually returned by a call to `mfactor` function.

OPTIONAL ARGUMENTS

variables : an integer vector telling which variables are to be plotted. The default is to plot all the variables, or the number of variables explaining 90% of the variance, whichever is bigger.

cumulative : a logical flag: if `TRUE`, the cumulative fraction of the variance is printed above each bar in the plot.

main : a character string giving the title of the plot. The default is `""`, which draws nothing in the title.

... : any optional argument that may be passed down to the plot function.

A screeplot of the object is drawn in a graphical device.

SEE ALSO

`plot.mfactor`,`screeplot` .

EXAMPLE

```
folio.mf = mfactor(folio.dat, 15)
screeplot(folio.mf)
```

SEMIFAR.fit Fitting of Semiparametric Fractional AR Models

DESCRIPTION
>   Called by the `SEMIFAR` function to fit the models. It is not supposed to be called by the users directly.

>   ```
>   SEMIFAR.fit(xInput.SEMI, p.range=c(0, 2), mmax=1, max.iter=20,
>               mse.RANGE=0.15, trace=T, ...)
>   ```

REQUIRED ARGUMENTS

`xInput.SEMI` : a numeric vector giving the time series to be modeled.

OPTIONAL ARGUMENTS

`p.range` : a vector with two positive integers that specifies the range for the order of the autoregressive part.

`mmax` : a positive integer that specifies the maximal number of difference to apply to the time series `xInput.SEMI` before fitting a stationary fractional AR model. Currently this must be either 0 or 1. If you only want to consider stationary models, set `mmax` to 0.

`mse.RANGE` : the IMSE is calculated by integrating the MSE over the interval `[mse.RANGE, 1-mse.RANGE]`.

`max.iter` : the maximal number of iterations for the semiparametric estimation of the smooth trend.

`trace` : a logical flag: if `TRUE`, a message will be printed on the screen when the function estimates the smooth trend iteratively.

`...` : any other optional argument that can be passed down to `FAR` function to control the estimation of fractional autoregressive part.

SEMIFAR.object Semiparametric Fractional AR Model Objects

DESCRIPTION

These are objects of class `"SEMIFAR"` and represent the fit of a semi-parametric fractional AR model.

GENERATION

This class of objects is returned from the `SEMIFAR` function.

METHODS

The `"SEMIFAR"` class of objects currently has methods for the following generic functions:

`coef`, `residuals`, `vcov`, `plot`, `predict`, `print`, `summary`.

STRUCTURE

The following components must be present in a legitimate `"SEMIFAR"` object:

`call` : an image of the call that produced the object.

`model` : a list with the following named components: `"d"` which is the estimated fractional difference parameter, `"ar"` is the estimated AR coefficients, and `"sigma2"` is the residual variance estimate.

`m` : an integer which is either 0 or 1. If `m=0`, the SEMIFAR model is fitted to the original time series; if `m=1`, the SEMIFAR model is fitted to the first difference of the original time series.

`delta` : a number between -0.5 and 0.5, which is the estimate of the fractional difference parameter after taking consideration of `m`. In general, the `"d"` component of `model` is equal to `m+delta`.

`BIC` : the Bayesian Information Criterion for the underlying fractional autoregressive model.

`loglike` : the log-likelihood value of the underlying fractional autoregressive model.

`residuals` : the residuals from the fitted model.

`cov` : the covariance matrix of the estimated model parameters.

`CI` : a matrix with two columns giving the confidence intervals of the estimated model parameters. This is returned only if `CI=T` in the original function call.

`trend` : the inferred nonparametric smooth trend.

`g.CI` : the 95% confidence interval of the smooth trend.

`bandwidth` : optimal bandwidth for the nonparametric estimation of smooth trend.

SEE ALSO
    `FAR, FARIMA ,SEMIFAR .`

SEMIFAR Fit a Semiparametric Fractional Autoregressive Model

DESCRIPTION

Fits a semiparametric fractional AR model to a univariate time series.

```
SEMIFAR(x, p.range=c(0, 2), mmax=1, mse.RANGE=0.15, max.iter=20,
        save.X=F, trace=T, ...)
```

REQUIRED ARGUMENTS

x : a numeric vector, or a `"timeSeries"` object with a numeric data slot.

OPTIONAL ARGUMENTS

p.range : a vector with two positive integers that specifies the range for the order of the autoregressive part.

mmax : a positive integer that specifies the maximal number of difference to apply to the time series x before fitting a stationary fractional AR model. Currently this must be either 0 or 1. If you only want to consider stationary models, set mmax to 0. The default is set to be 1.

mse.RANGE : the IMSE is calculated by integrating the MSE over the interval [mse.RANGE, 1-mse.RANGE].

max.iter : the maximal number of iterations for the semiparametric estimation of the smooth trend.

save.X : a logical flag: if TRUE, the original time series x will be saved in the returned object. The default is FALSE.

trace : a logical flag: if TRUE, a message will be printed on the screen when the function estimates the smooth trend iteratively.

... : any other optional argument that can be passed down to FAR function to control the estimation of fractional autoregressive part.

VALUE

an object of class `"SEMIFAR"` representing the fit of the semiparametric fractional AR model. See `SEMIFAR.object` for details.

REFERENCES

Beran, J. (1995). Maximum likelihood estimation of the differencing parameter for invertible short and long memory ARIMA models. *Journal of Royal Statistical Society B*, 57(4):659-672.

Beran, J., Feng, Y., and Ocker, D. (1999). SEMIFAR models. *Technical Report*, 3/1999, SFB 475 University of Dortmund.

Beran, J., and Ocker, D. (2001). Volatility of stock market indices - an analysis based on SEMIFAR models. *Journal of Business and Economic Statistics*, 19(1):103-116.

SEE ALSO

`arima.fracdiff`, `FAR` ,`FARIMA` ,`SEMIFAR.object` .

EXAMPLE

```
set.seed(2)
ts.sim <- simulate.FARIMA(list(d=.3, ar=.5, ma=0.1, mean=1), n=3000)
SEMIFAR(ts.sim, mmax=0)
```

shape.plot   Calculate and Plot GPD Shape Parameter

DESCRIPTION

Calculates and generates a plot showing how the estimate of GPD shape parameter varies with thresholds.

```
shape.plot(data, method="ml", from=0.5, to=0.98, nint=30)
```

REQUIRED ARGUMENTS

`data:` a numeric vector or `"timeSeries"` object.

OPTIONAL ARGUMENTS

`ml:` a character string specifying the estimation method to fit GPD distribution: `"lmom"` or `"ml"`. The default is `"ml"`.

`from:` an integer specifying the beginning percentile of the data to use as threshold to fit the models. The default is `0.5`.

`to:` an integer specifying the ending percentile of the data to use as threshold to fit the models. The default is `0.98`.

`nint:` a numerical value specifying the number of GPD models to fit. The default is `30`.

A plot of estimated GPD shape parameters over increasing thresholds will be drawn on a graphical device.

SEE ALSO

`shape`, `gpd.lmom` , `gpd.ml` .

EXAMPLE

```
shape.plot(UTI.GAS[,2])
```

shape Calculate and Plot GPD Shape Parameter

DESCRIPTION

Calculates and generates a plot showing how the estimate of GPD shape parameter varies with threshold or number of extremes.

```
shape(data, models=30, start=15, end=500, reverse=T, ci=0.95,
      auto.scale=T, labels=T, table=F, ...)
```

REQUIRED ARGUMENTS

data: a numeric vector or `"timeSeries"` object.

OPTIONAL ARGUMENTS

models: an integer specifying the number of consecutive GPD models to be fitted. The default is 30.

start: an integer specifying the lowest number of exceedances in a series of GPD models to fit. The default is 15.

end: an integer specifying the maximum number of exceedances in a series of GPD models to fit. The default is 500.

reverse: a logical flag: if `TRUE` plot is to be by increasing threshold; if `FALSE` by increasing number of extremes. The default is `TRUE`.

ci: a number between 0 and 1 specifying the probability for asymptotic confidence band. For no confidence band set `ci` to `FALSE`. The default is 0.95.

auto.scale: a logical flag: if `TRUE`, plot is automatically scaled; if `FALSE`, `xlim` and `ylim` graphical parameters may be entered. The default is `TRUE`.

labels: a logical flag: if `TRUE`, axes are labeled. Labels are ignored if `FALSE`. The default is `TRUE`.

table: a logical flag: if `TRUE`, a table of results is printed. The default is `FALSE`.

... : any optional arguments that can be passed down to the `plot` function.

VALUE

a matrix which contains the following columes:

threshold : the thresholds.

shape : the shape parameters of fitted GPD distribution.

se : the standard errors of the shape parameter.

> `exceedances` : the number of extremes. A plot of estimated GPD shape parameters over increasing thresholds or number of extremes will be drawn on a graphical device.

SEE ALSO

> `shape.plot`, `gpd` , `plot.gpd` , `hill` .

EXAMPLE

> ```
> # Shape plot of heavy-tailed Danish fire insurance data
> shape(danish)
> ```

SHAPE.XI SHAPE.XI - Parameterization of GEV and GPD

DESCRIPTION

      For financial applications, it is customary to define GEV and GPD distribution by a shape parameter `xi`. However, for other applications, e.g. hydrology, it is customary to use another parameterization of the GEV and GPD by defining a new shape parameter `k=-xi`. Thus, the global variable `SHAPE.XI` is to set to `TRUE` for financial applications. For hydrology applications, it is set to be `FALSE`.

      User may choose to work with the parameterization he or she is more comfortable with. The default value is `TRUE`.

DETAILS

      Setting `SHAPE.XI` affects the parameterization and return values of following functions:

      `gev.lmom`, `gev.mix1`, `gev.mix2`, `sample.LMOM`, `PlotPos.LMOM`, `gpd.lmom`, `gpd.ml`, `gpd.tail`, `gpd.1p`, `gpd.2p`, `gpd.1q`, `gpd.2q`.

REFERENCES

      Leadbetter, M. R., Lindren, G., and Rootzen, H. (1983). *Extremes and Related Properties of Random Sequences and Processes*. New York: Springer-Verlang.

      Embrechts, P., Kluppelberg, C., and Mikosch, T. (1997). *Modelling Extremal Events for Insurance and Finance*. New York: Springer-Verlag.

shiller.dat Data Sets Used in Irrational Exuberance by Robert Shiller

SUMMARY

These are `"timeSeries"` objects representing some data used in the book Irrational Exuberance by Robert Shiller. The data in `shiller.dat` are monthly stock price, dividends, earnings, etc, starting from January 1871 to March 2001, while the data in `shiller.annual` are the annual figures of the corresponding variables. They have the following columns:

`price:` nominal U.S. stock market price, based on Standard and Poor's Statistical Service Security Price Index Record.

`dividend:` nominal Standard and Poor Composite Index dividend.

`earnings:` nominal Standard and Poor Composite Index earnings.

`cpi:` U.S. Consumer Price Index.

`real.price:` real U.S. stock market price.

`real.dividend:` real Standard and Poor Composite Index dividend.

`real.earnings:` real Standard and Poor Composite Index earnings.

`pe.10:` price-earnings ratio.

`dp.ratio:` dividend-price ratio. This column is only present in `shiller.annual`.

`dp.yield:` dividend-price yield. This column is only present in `shiller.annual`.

SOURCE

Shiller, R. J. (1989). *Market Volatility*. MIT Press.

Shiller, R. J. (2001). *Irrational Exuberance*. Broadway Books.

siemens Daily Log Returns on Siemens Share Price

SUMMARY

This is the daily `"timeSeries"` object for log returns of Siemens share price from 1/2/1973 to 7/23/1996. Note no trading takes place at the weekend.

sigma.t Extract Conditional Standard Deviation

DESCRIPTION

Extracts conditional standard deviations from an object of class inheriting `garch` or `mgarch`.

`sigma.t(x)`

REQUIRED ARGUMENTS

x: an object of class inheriting `"garch"` or `"mgarch"`.

VALUE

a vector of the conditional standard deviation series for the univariate case, or a matrix with each row being the square roots of the diagonal elements of the conditional covariance matrices for the multivariate case.

SEE ALSO

`garch`, `fgarch` , `mgarch` .

SimSmoDraw Simulation Smoother Draws

DESCRIPTION

Generates random sample draws from disturbance simulation smoother or state simulation smoother.

```
SimSmoDraw(kf, ssf, task="DSSIM", mRan=NULL, a1=NULL)
```

REQUIRED ARGUMENTS

kf : an object of class `"KalmanFil"` as returned by a call to `KalmanFil`.

ssf : a list which either contains the minimal necessary components for a state space form or is a valid `"ssf"` object.

OPTIONAL ARGUMENTS

task : a character string which specifies the simulation smoother to use. Valid choices are: `"DSSIM"` for disturbance simulation, and `"STSIM"` for state simulation. The default is to use disturbance simulation smoother.

mRan : a `cT x (cSt+cY)` matrix which represents the disturbance of the state space model, where `cT` is the number of sample observations in the original data used in `kf`, `cSt` is the number of state variables, and `cY` the number of response variables or observables. If `NULL`, a random draw from the disturbance distribution is generated automatically. The default is `NULL`.

a1 : the values of state variables to start off the state space recursion. If `NULL`, the initial mean of the state variables is used. The default is `NULL`.

VALUE

an S version 3 object of class `"SimSmoDraw"` which contains the following components:

state: a numeric vector or matrix which represents the simulation smoother draws of the state variables or disturbance.

response: a numeric vector or matrix which represents the simulation smoother draws of the response variables or disturbance.

task: a character string which is the same as the input `task`. If `mRan` is `NULL`, then the `.Random.seed` object is updated in current working directory.

DETAILS

Currently there is only a `plot` method for objects of class `"SimSmoDraw"`.

For state simulation smoother, the `kf` object must be generated by a call to `KalmanFil` with `task="STSIM"`.

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Durbin, J., and Koopman, S. J. (2001). A simple and efficient simulation smoother for state space time series analysis. mimeo.

SEE ALSO

`CheckSsf`,`SsfCondDens` ,`SsfSim` .

EXAMPLE

```
ssf.mod = GetSsfStsm(irregular=sqrt(15099), level=sqrt(1469))
nile.fil = KalmanFil(nile.dat, ssf.mod)
nile.draw = SimSmoDraw(nile.fil, ssf.mod, task="DSSIM")
```

simulate.FARIMA Simulation of a Fractional ARIMA Model

DESCRIPTION
>Generates simulations from a given fractional ARIMA model.

>```
>simulate.FARIMA(model, n=10000)
>```

REQUIRED ARGUMENTS

`model` : a list with the following named components: `"d"` which is the fractional difference parameter, `"ar"` which is the AR coefficients, `"ma"` which is the MA coefficients, `"mean"` which is the unconditional mean of the model, and `"sigma2"` which is the variance of the residual term. If `model` has class `"FARIMA"`, its `model` component will be taken.

OPTIONAL ARGUMENTS

>`n` : an integer specifying the number of observations to simulate from the given model.

VALUE
>a numeric vector with length `n` which follow the FARIMA model given in `model`. Sets the value of the `.Random.seed` object in the working directory.

DETAILS
>This function uses the same algorithm as in `arima.fracdiff.sim` .

SEE ALSO
>`FARIMA.object`,`arima.sim`,`arima.fracdiff.sim`,`simulate.garch`,`simulate.mgarch` ,`simulate.VAR` .

EXAMPLE
>```
># simulates a fractional Gaussian noise
>simulate.FARIMA(list(d=0.3), 1000)
>```

simulate.garch Function to Generate Univariate GARCH Series

DESCRIPTION

Simulates a time series from a user specified univariate GARCH model. The types of univariate GARCH series include not only the basic GARCH model, but also the following generalizations: `pgarch`, `tgarch`, `egarch`, and `two.comp` models. Simulation of a time series for the conditional mean is not provided by this function.


```
simulate.garch(model, sigma=T, n=1000, n.start=1000,
    n.rep=100, cond.dist="Gaussian", etat=NULL,
    dist.par=NULL, sigma.start=NULL, rseed=NULL)
```

REQUIRED ARGUMENTS

  model : the argument `model` can be a list with the model parameters as components. It can also be an object of class `"garch"` or `"garch.model"`. When the model is specified by a list, the following components must be included:

  egarch : logical flag: if `TRUE`, simulates an exponential garch model (EGARCH).

  tgarch : logical flag: if `TRUE`, simulates a threshold GARCH model (TGARCH).

  a.value : the constant term in the variance equation.

    arch : a vector giving the moving average coefficients of the model.

   garch : a vector giving the autoregression coefficients of the model.

   power : a numeric variable, specifying the power in a power GARCH model.

  lev.value : a vector giving the leverage coefficients of a GARCH model with leverage terms.

OPTIONAL ARGUMENTS

  sigma : logical flag: if `TRUE`, the generalized conditional standard deviation sequence will be provided as a component of the output list.

      n : the length of the output series.

  n.start : the length of the warm-up sequence to reduce the effect of initial conditions.

  n.rep : the number of times to replicate the simulation. The default is set to 1.

  cond.dist : a character string giving the name of the conditional distribution. The options are `"Gaussian"`, `"t"`, `"double.exp"`, and `"ged"`.

etat: a vector or matrix of residuals to use to generate the GARCH series. The default is to use a vector of standard normal errors.

dist.par: the parameters of the conditional distribution.

sigma.start: the initial conditional variance sequence to start the iterations.

rseed: a positive integer used to set a specific seed for the random number generator. The value is used in a call to set.seed.

VALUE

a vector representing the simulated GARCH process if `sigma=F`, or a list with the following two components if `sigma=T`:

et: the generated GARCH process.

sigma.t : the corresponding conditional standard deviation sequence.

The model specified by the user appears as an attribute of the list. The dataset .Random.seed is created if it does not already exist, otherwise its value is updated.

DETAILS

It is important to specify "stable" values of model parameters which yield a stationary time series to prevent numeric overflow.

REFERENCES

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50:987-1006.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroscedasticity. *Journal of Econometrics*, 31:307-327.

SEE ALSO

simulate.mgarch, arima.sim .

EXAMPLE

```
# Generate an ARCH(1) series of length 500:
et = simulate.garch(model=list(a.value=0.1, arch=0.8),
    n=500, n.start=100,  rseed=196, sigma=F)


# Generate a GARCH(2,2) series by a model list:
model.g22 = list(a.value=0.1,arch=c(0.6,0.1), garch=c(0.1,0.1))
et = simulate.garch(model=model.g22)
```

```
# Generate a PGARCH(2,2) series with leverage terms:
model.pgl22 = list(a.value=0.1, arch=c(0.6,0.1), garch=c(0.1,0.1),
                   leverage=c(-0.1, -0.1), power=1.0)
et = simulate.garch(model=model.pgl22)
```

simulate.mgarch Generate Multivariate GARCH Series

DESCRIPTION

Simulates one of a wide variety of multivariate GARCH models, including not only the basic diagonal vec models of Bollerslev, Engle, and Nelson (1986), but also the complete set of models given in Ding (1994). However only order(1,1) models are supported.

```
simulate.mgarch(model=list(A=matrix(c(0.5,0.1,0.1,0.4), 2, 2),
    B=matrix(c(0.35,0.22,0.22,0.12), 2, 2),
    C=matrix(c(0.2,0.1,0.1,0.3), 2, 2), y.dim=2, idmod=101),
    n=1000, n.start=1000, etat=NULL, V.start=NULL, rseed=NULL)
```

REQUIRED ARGUMENTS

model: can be an object of class `"mgarch"`, or `"mgarch.model"`, or a list specifying the model to be simulated. The following components should be given for a pure list:

A: a matrix, a vector, or a scalar used in the conditional variance equation. For the principal component model, `A` gives the matrix **P** in the conditional variance equation (symmetric); for the constant conditional correlation model, `A` specifies the correlation matrix **R** in the conditional variance equation.

B: a matrix, a vector, or a scalar used in the conditional variance equation.

C: a matrix used in the conditional variance equation.

idmod: a numeric variable giving the model identification number.

y.dim: a numeric variable specifying the dimension of vector time series.

OPTIONAL ARGUMENTS

n: the length of the multivariate time series simulated.

n.start: the length of the vector series to warm up the iterations to minimize the effects of initial conditions.

etat: a matrix of residuals to use to generate the multivariate GARCH series, with the first row used as the pre-sample residuals. The default is to use a standard normal errors.

V.start: a numeric matrix giving the initial value of the conditional covariance matrix. The default is `NULL`, indicating use of the identity matrix as the initial value.

rseed: a positive integer used to set a specific seed for the random number generator. The value is used in a call to `set.seed`.

VALUE

a list containing the following two components:

et: a matrix of dimension `n x y.dim`. Each column gives a generated GARCH time series.

V.t: an array of dimension `y.dim x y.dim x n` giving the corresponding conditional covariance matrices for DVEC and BEKK models, or a matrix of dimension `n x y.dim` giving the conditional standard deviations for pure diagonal models. The dataset `.Random.seed` is created if it does not already exist, otherwise its value is updated.

DETAILS

It is important to specify "stable" values of `A`, `B` and `C`, i.e., parameter values that yield a stationary multivariate time series to prevent numeric overflow. Any singularity in the conditional variance matrix sequence will be reported by the Cholesky decomposition used by `simulate.mgarch`.

REFERENCES

Bollerslev, T., Engle, R. F. and Nelson, D. B. (1994). ARCH models. In *Handbook of Econometrics, Vol. IV*, Engle and McFadden, eds., Elsevces Science B.V. pp. 2959-3038,

Ding, Z (1994). Time Series Analysis of Speculative Returns. *Ph.D. dissertation,* Dept. of Economics, University of California, San Diego.

SEE ALSO

`simulate.garch`, `mgarch` .

EXAMPLE

```
# A BEKK model:
a = matrix(c(0.5, 0.1, 0.1, 0.4), 2, 2)
b = matrix(c(0.35, 0.22, 0.22, 0.12), 2, 2)
c0 = matrix(c(0.34, 0.17, 0.0, 0.23), 2, 2)
model.bekk = list(A=a, B=b, C=c0, idmod=22, y.dim=2)
et = simulate.mgarch(model=model.bekk, n=500, n.start=200)
```

simulate.VAR Vector Autoregressive Model Simulation

DESCRIPTION

Simulates a multivariate time series from a fitted or user specified vector autoregressive (VAR) model.

```
simulate.VAR(object, n=100, n.start=0, n.rep=1, etat=NULL,
             y0=NULL, unbiased=T)
```

REQUIRED ARGUMENTS

object : an object which inherits the class `"VAR"`, or a model list with the following components: `const` which is the constant term in the VAR model, `ar` which gives the autoregressive coefficient matrix, and `Sigma` which gives the error covariance matrix. When `object` is a model list, then the `ar` component must follow the format of the `coef` component of a `"VAR"` object, that is, equations are represented by columns in the `ar` component.

OPTIONAL ARGUMENTS

n : the required length of the multivariate time series to be generated. The default is set to 100.

n.start : the required length of the warm-up sequence to reduce the effect of initial conditions. The default is set to 0.

n.rep : the number of times to replicate the simulation, or the number of sample paths to simulate. The default is set to 1.

etat : a matrix of residuals or error terms to use to simulate the VAR model. When supplied, it should be an array of dimension (n + n.start) x k x n.rep, where k is the dimension of the multivariate time series to be generated. When n.rep=1, it can be a matrix of dimension (n + n.start) x k. The default is set to NULL, which generates multivariate normal errors using the given error covariance matrix.

y0 : the pre-sample values to start the simulation. If supplied, it must be a matrix with dimension p x k, where p is the order of the autoregressive model, and k is the dimension of the multivariate time series. By default, zeros are used when `object` is a model list, and the last p observations in the original data are used if `object` inherits from class `"VAR"`.

unbiased : a logical flag: if TRUE, then unbiased error covariance matrix is used. This is only used when `object` inherits from the class `"VAR"`, and ignored otherwise. The default is set to TRUE.

VALUE

a matrix with dimension n x k representing the simulated VAR series if n.rep=1, or an array with dimension n x k x n.rep if n.rep is greater than 1. The dataset .Random.seed is created if it does not already exist, otherwise its value is updated.

SEE ALSO

arima.sim,simulate.garch ,simulate.mgarch ,simulate.FARIMA .

EXAMPLE

```
# simulation forecasts from a fitted VAR
sam.mod = VAR(cbind(FFR, U)~ar(3), data=policy.dat)
sim.pred = simulate(sam.mod, n=10, n.rep=1000)
apply(sim.pred, 2, rowMeans)
```

singleIndex.dat Microsoft Stock Price and S & P 500 Index

SUMMARY
These are monthly `"timeSeries"` objects from January 1990 to January 2001. `singleIndex.dat` contains the monthly closing prices for Microsoft Corporation and S & P 500 index, while `excessReturns.ts` contains the corresponding monthly returns in excess of the risk free rate, adjusted for dividends and stock splits. Both have two columns:

`MSFT:` the closing price or excess return for Microsoft Corporation.

`SP500:` the closing price or excess return for S & P 500 index.

SMA Simple Moving Average/Rolling Sample Mean

DESCRIPTION

Returns the simple moving average, or equivalently, the rolling sample mean, of a given time series.

```
SMA(x, n=9, trim=T, na.rm=F)
```

REQUIRED ARGUMENTS

x : a vector, matrix, or a `"timeSeries"` object.

OPTIONAL ARGUMENTS

n : a positive integer that determines the number of periods to use in the moving average.

trim : a logical flag: if `TRUE`, the first `n-1` missing values in the output are trimmed; if `FALSE`, they are saved in the returned value.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computing SMA. The default is `FALSE`.

VALUE

a vector, matrix, or a `"timeSeries"` object representing the simple moving average of x. If `trim=F`, it has the same dimension as x; otherwise, it starts from the n-th observation.

DETAILS

If there are more than one column in x, then SMAs are computed column-wise.

SEE ALSO

EWMA,iEMA ,iMA .

EXAMPLE

```
bond = nelson.dat[positions(nelson.dat) > timeDate("01/01/1899"), "BND"]
plot(bond, sma(bond, n=5))
```

sp500 S & P 500 Returns

SUMMARY

The `sp500` data vector is a data set with 17054 values, representing S&P 500 returns from January 4, 1928, to August 30, 1991. The time series `sp500.s` is the last 2000 data points from `sp500`.

Spearmans.rho   Spearman's rho of a Copula

DESCRIPTION
   Computes Spearman's rho for a parametric copula or an empirical copula.

   `Spearmans.rho(copula, tol=1e-5)`

REQUIRED ARGUMENTS

`copula:` an object of class `"copula"`.

  `tol:` a numerical value defining the absolute tolerance for the integral if numerical integration is required. The default is 10^{-5}.

VALUE
   Spearman's rho for the copula.

DETAILS
   This is a generic function.   All copula classes (including empirical copula) have their method functions.

REFERENCES
   Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO
   `copula.object, Kendalls.tau` .

EXAMPLE
```
# This example compares the Spearman's rho of an empirical copula
# with that of the parametric copula
# that was used for its generation
ec = empirical.copula(rcopula(bb5.copula(1.5,0.7), 1000))
Spearmans.rho(ec)
Spearmans.rho(bb5.copula(1.5,0.7))
```

sp.raw Daily SP500 Index

SUMMARY

This is the daily `"timeSeries"` object for closing values of the SP500
index from 1/4/1960 to 10/16/1987.

SsfCondDens   Conditional Density/Mean Calculation of State Space Models

DESCRIPTION

Returns the expectations or smoothed values of the state variables, response variables, or disturbance of a state space model, given observations of the response variables or signals.

```
SsfCondDens(mY, ssf, task="STSMO", ikf=NULL)
```

REQUIRED ARGUMENTS

mY : a rectangular numeric object which represents the response variables or the observables. Note that for multivariate response, the variables must be in columns.

ssf : a list which either contains the minimal necessary components for a state space form or is a valid `"ssf"` object.

OPTIONAL ARGUMENTS

task : a character string which specifies the task to be performed. Valid choices are: `"DSSMO"` for disturbance smoothing, and `"STSMO"` for state smoothing. The default is to perform state smoothing.

ikf : an object returned by a call to `KalmanIni` function. If NULL, it will be constructed by `SsfCondDens` automatically. If the same call to `SsfCondDens` has to be made repeatedly, it is more efficient to construct and pass `ikf` before calling `SsfCondDens` function.

VALUE

an object of class `"SsfCondDens"` which contains the following components:

state: a numeric vector or matrix which represents the conditional mean of the state variables or disturbance.

response: a numeric vector or matrix which represents the conditional mean of the response variables or disturbances.

task: a character string which is the same as the input `task`.

positions: a `"timeDate"` vector giving the positions of the input data `mY`. This is only returned if `mY` is a `"timeSeries"` object.

DETAILS

Currently there is only a `plot` method for objects of class `"SsfCondDens"`.

The `SsfCondDens` function only returns the conditional mean of the required variables. To obtain the conditional variance, use `SsfMomentEst`.

In contrast, the function `SimSmoDraw` generates random samples from the same conditional density.

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Koopman, S. J., Shephard, N., and Doornik, J. A. (1999). Statistical algorithms for methods in state space form using SsfPack 2.2. *Econometric Journal*, 2:113-166.

SEE ALSO

CheckSsf,KalmanIni ,SimSmoDraw ,SsfMomentEst .

EXAMPLE

```
ssf.mod = GetSsfStsm(irregular=sqrt(15099), level=sqrt(1469))
stsm.ikf = KalmanIni(nile.dat, ssf.mod, task="STSMO")
SsfCondDens(nile.dat, ssf.mod, ikf=stsm.ikf, task="STSMO")
```

SsfFit Estimate Unknown Parameters of a State Space Model

DESCRIPTION

Returns maximum likelihood estimates of the unknown parameters of a state space model using prediction error decomposition.

```
SsfFit(parm, data, FUN, conc=F, scale=1, gradient=NULL,
        hessian=NULL, lower=-Inf, upper=Inf, trace=T,
        control=NULL, ...)
```

REQUIRED ARGUMENTS

parm : a numeric vector which gives the starting values of the unknown parameters for maximum likelihood estimation.

data : a rectangular numeric object which represents the response variables or the observables. Note that for multivariate response, the variables must be in columns.

FUN : a character string giving the name of the function which takes `parm` together with the optional arguments in ... and produces an `"ssf"` object representing the state space form.

OPTIONAL ARGUMENTS

conc : a logical flag: if `TRUE`, the concentrated log-likelihood value will be used. The default is `FALSE`.

scale : either a single positive value or a numeric vector with positive components of length equal to the length of `parm` to be used to scale the parameter vector. See the help file for `nlminb` for more details. The default is 1.

gradient : vector-valued function that computes the gradient of the log-likelihood function of the state space model. See the help file for `nlminb` for details. If `NULL`, numeric gradients are used. The default is `NULL`.

hessian : either a vector-valued function that computes the entries of the Hessian matrix of the log-likelihood function of the state space model, or a logical variable indicating whether or not the Hessian computation takes place within `gradient`. If `NULL`, numeric Hessian is used. The default is `NULL`.

lower, upper : either a single numeric value or a vector of length equal to the length of `parm` giving lower or upper bounds for the parameter values. The absence of a bound may be indicated by either `NA` or `NULL`, or by `-Inf` and `Inf`. The default is unconstrained minimization: `lower=-Inf` and `upper=Inf`.

trace : a logical flag: if `TRUE`, a message will be printed on screen giving the log-likelihood value for each iteration. The default is `TRUE`.

control : a list of parameters by which the user can control various aspects of the minimization. For details, see the help file for `nlminb.control`.

... : additional arguments that are passed to `FUN` to return the state space form.

VALUE

a list with the same components as returned by `nlminb`. See the help file for `nlminb` for details.

DETAILS

`SsfFit` calls `nlminb` to perform maximum likelihood estimation.

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

SEE ALSO

`nlminb,SsfLoglike` .

EXAMPLE

```
local.level = function(parm)

  parm = exp(parm)
  ssf.mod = GetSsfStsm(irregular=sqrt(parm[1]),
                        level=sqrt(parm[2]))
  CheckSsf(ssf.mod)

# estimate the log variances of a local level model
nile.start = c(9, 7)
names(nile.start) = c("irregular", "level")
SsfFit(nile.start, nile.dat, "local.level")$parameters
```

SsfLoglike   Log-likelihood Value of a State Space Model

DESCRIPTION

Returns the log-likelihood value of a state space model using prediction error decomposition.

```
SsfLoglike(mY, ssf, scores=F, ikf=NULL)
```

REQUIRED ARGUMENTS

mY : a rectangular numeric object which represents the response variables or the observables. Note that for multivariate response, the variables must be in columns.

ssf : a list which either contains the minimal necessary components for a state space form or is a valid `"ssf"` object.

OPTIONAL ARGUMENTS

scores : a logical flag: if `TRUE`, the analytical scores for the parameters in the disturbance covariance will also be computed. The default is `FALSE`.

ikf : an object returned by a call to `KalmanIni` function. If `NULL`, it will be constructed by `SsfLoglike` automatically. If the same call to `SsfLoglike` has to be made repeatedly, it is more efficient to construct and pass `ikf` before calling `SsfLoglike` function.

VALUE

a list with the following components:

loglike: the log-likelihood value of the state space model.

loglike.conc: the concentrated log-likelihood value of the state space model.

err: a logical flag: if `TRUE`, the underlying Kalman filtering was successful; if `FALSE`, the underlying Kalman filtering failed.

scores: a numeric matrix representing the analytical scores for the parameters in disturbance covariance matrix.

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Koopman, S. J., Shephard, N., and Doornik, J. A. (1999). Statistical algorithms for methods in state space form using SsfPack 2.2. *Econometric Journal*, 2:113-166.

SEE ALSO

    CheckSsf,KalmanFil ,KalmanIni ,SsfFit .

EXAMPLE

    ssf.mod = GetSsfStsm(irregular=sqrt(15099), level=sqrt(1469))
    SsfLoglike(nile.dat, ssf.mod, scores=T)

SsfMomentEst State Space Moment Filtering and Smoothing

DESCRIPTION

Returns the mean and variance of filtered or smoothed variables.

```
SsfMomentEst(mY, ssf, task="STSMO", ikf=NULL)
```

REQUIRED ARGUMENTS

mY : a rectangular numeric object which represents the response variables or the observables. Note that for multivariate response, the variables must be in columns.

ssf : a list which either contains the minimal necessary components for a state space form or is a valid `"ssf"` object.

OPTIONAL ARGUMENTS

task : a character string which specifies the purpose of the call to `SsfMomentEst` function. The default is `"STSMO"`, which returns smoothed state variables and response variables and their variance. Other valid choices are: `"STPRED"` for state and response prediction, `"STFIL"` for state and response filtering, `"DSSMO"` for disturbance smoothing.

VALUE

an S version 3 object of class `"SsfMomentEst"` which contains the following components:

state.moment: a numeric vector or matrix which represents the required moment in the state equation.

state.variance: the variance of `state.moment`.

response.moment: a numeric vector or matrix which represents the required moment in the measurement equation.

response.variance: the variance of `response.moment`.

task: a character string which is the same as the input `task`.

positions: a `"timeDate"` vector giving the positions of the input data `mY`. This is only returned if `mY` is a `"timeSeries"` object.

DETAILS

Currently there is only a `plot` method for objects of class `"SsfMomentEst"`.

REFERENCES

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford University Press.

Koopman, S. J., Shephard, N., and Doornik, J. A. (1999). Statistical algorithms for methods in state space form using SsfPack 2.2. *Econometric Journal*, 2:113-166.

SEE ALSO

CheckSsf,KalmanIni ,SimSmoDraw ,SsfCondDens .

EXAMPLE

```
# compute the smoothed disturbance
ssf.mod = GetSsfStsm(irregular=sqrt(15099), level=sqrt(1469))
SsfMomentEst(nile.dat, ssf.mod, task="DSSMO")
```

SsfSim State Space Simulation

DESCRIPTION
>   Generates a random sample from a state space form.

>   SsfSim(ssf, n=100, mRan=NULL, a1=NULL)

REQUIRED ARGUMENTS

>   ssf : a list which either contains the minimal necessary components for a
>   state space form or is a valid `"ssf"` object.

OPTIONAL ARGUMENTS

>   n : the length of the random sample to be generated. This is used when
>   `mRan` is not supplied. When `mRan` is supplied, `n` is ignored. The default
>   is set to 100.

>   mRan : a `cT x (cSt+cY)` matrix which represents the disturbance of the state
>   space model, where `cT` is the number of sample observations, `cSt` is the
>   number of state variables, and `cY` the number of response variables or
>   observables. If `NULL`, a random draw from the disturbance distribution
>   is generated automatically. The default is `NULL`.

>   a1 : the values of state variables to start off the state space recursion. If
>   `NULL`, the initial mean of the state variables is used. The default is
>   `NULL`.

VALUE
>   a `cT x (cSt+cY)` matrix which represents a random draw from the
>   state space form. Note that the first `cSt` columns contain the state
>   variables, while the last `cY` columns contain the response variables.
>   If `mRan` is `NULL`, then the `.Random.seed` object is updated in current
>   working directory.

REFERENCES
>   Koopman, S. J., Shephard, N., and Doornik, J. A. (1999). Statis-
>   tical algorithms for methods in state space form using SsfPack 2.2.
>   *Econometric Journal*, 2:113-166.

SEE ALSO
>   **CheckSsf**,**SimSmoDraw** .

EXAMPLE
```
ssf.mod = GetSsfStsm(irregular=sqrt(15099), level=sqrt(1469))
set.seed(1)
SsfSim(ssf.mod, n=100, a1=1120)
```

stationaryTest   Test for Stationarity Against Unit Root or Long Memory

DESCRIPTION

Returns the KPSS test for stationarity against unit root or long memory.

```
stationaryTest(x, trend="c", bandwidth=NULL, na.rm=F)
```

REQUIRED ARGUMENTS

x : a vector, matrix, or a `"timeSereis"` object with a numeric object in the data slot.

OPTIONAL ARGUMENTS

trend : a character string that gives the trend specification for the underlying regression. Valid choices are: `"c"` for constant only, and `"ct"` for constant and time trend. The default is `"c"`.

bandwidth : an integer that specifies the bandwidth to be used. If it is 0, then sample variance is used. Otherwise, the Newey-West long run variance is used. The default is set to `4*(n/100)^0.25`, where `n` is the sample size of `x`.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computing the test statistics. The default is `FALSE`.

VALUE

an object of class `"stationaryTest"`, which contains the following components:

trend : a character string which is the same as in the input `trend` argument.

stat : a numeric vector giving the KPSS test statistics.

bandwidth : an integer giving the chosen bandwidth used in the estimation of long run variance.

n : an integer giving the sample size.

na : an integer giving the number of missing values in the input if any.

REFERENCES

Lee, D., and Schmidt, P. (1996). On the power of the KPSS test of stationarity against fractionally-integrated alternatives. *Journal of Econometrics*, 73:285-302.

Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., and Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, 54:159-178.

SEE ALSO

gphTest,rosTest ,unitroot .

EXAMPLE

stationaryTest(log(nelson.dat[,"WG"]), trend="ct", na.rm=T)

stocks New York Stock Exchange Data

SUMMARY

The `stocks` data matrix is a data set with 5 columns and 7420 rows. The column names are `AMOCO`; `FORD`; `HP`; `IBM`; `MERCK`. Each series is the daily stock returns from July 3, 1962, to December 31, 1991 (7420 trading days).

DATA DESCRIPTION

This data matrix contains the following columns:

`AMOCO`: daily returns of AMOCO company.

`FORD`: daily returns of FORD company.

`HP`: daily returns of HP company.

`IBM`: daily returns of IBM company.

`MERCK`: daily returns of MERCK company.

The individual time series `amoco.s`, `ford.s`, `hp.s`, `ibm.s`, and `merck.s` are the last 2000 data points from `stocks`, representing the daily returns for the five companies from Feburary 2, 1984, to December 31, 1991.

summary.arima.rob Summary Method for `arima.rob` Objects

DESCRIPTION
Returns a summary list for an `"arima.rob` object.

        summary.arima.rob(object, corr=F)

REQUIRED ARGUMENTS

object: an object of class `"arima.rob"`.

OPTIONAL ARGUMENTS

corr: a logical flag: if `TRUE`, correlation matrices of regression coefficients and ARIMA coefficients are also produced. The default is `FALSE`.

VALUE
an object of class `"summary.arima.rob"` which must contain the following components:

ARIMA.model: the same list as the `model` component of `object`. See `arima.rob.object` for details.

reg.coef: a matrix with four columns, containing the regression coefficients, their standard errors, the t-statistics and the corresponding p-values.

regcoef.cov: the estimated covariance matrix for the regression coefficients.

regcoef.corr: the estimated correlation matrix for the regression coefficients. This is only present if `corr=T`.

AR.coef: a matrix with four columns, containing the AR coefficients, their standard errors, the t-statistics and the p-values.

MA.coef: a matrix with four columns, containing the MA coefficients, their standard errors, the t-statistics and the p-values.

sMA.coef: an array which contains the seasonal moving average parameter, its standard error, the t-statistic and the p-value.

ARIMA.cov: the estimated covariance matrix of the ARMA coefficients.

ARIMA.corr: the estimated correlation matrix of the ARMA coefficients. This is only present if `corr=T`.

n: the length of the time series.

df: the number of degrees of freedom for the model.

sigma: the estimate of the innovations scale.

`call:` the image of the original call to `arima.rob`.

`outliers:` an object of class `"summary.outliers"`.

DETAILS

This function is a method for the generic function `summary` for class `"arima.rob"` . It can be invoked by calling `summary` for an object of the appropriate class, or directly by calling `summary.arima.rob` regardless of the class of the object.

SEE ALSO

`arima.rob`,`arima.rob.object` ,`summary` .

EXAMPLE

```
frip.rr = arima.rob(log(frip.dat)~1, p=2, d=1)
summary(frip.rr)
```

summary.FARIMA Summary Method for FARIMA Models

DESCRIPTION

Returns some summary statistics for a fractional ARIMA model. A null value will be returned if printing is invoked.

```
summary.FARIMA(object)
```

REQUIRED ARGUMENTS

object: an object inheriting from class `"FARIMA"`, usually returned by `FARIMA` function.

VALUE

an object of class `"summary.FARIMA"`, which is the same as `object` with the following change:

coef: a matrix with four columns, containing the coefficient estimates, their standard errors, the corresponding t statistics and p-values.

SEE ALSO

`FARIMA.object`, `summary` .

summary.fgarch Summary for an Object of Fitted FGARCH Object

DESCRIPTION
  Summarizes an object of class `"fgarch"`.

  `summary.fgarch(x, max.lag=12, method="OP")`

REQUIRED ARGUMENTS

  x: an object of class of `"fgarch"`, usually an object generated by the
  function `fgarch`.

OPTIONAL ARGUMENTS

`max.lag`: maximum number of lags used in Lagrange Multiplier test and Ljung-
  Box test.

`method`: a character string specifying the type of covariance matrix to be used.
  Valid choices are `"OP"` for outer product version of the covariance ma-
  trix, `"HESSIAN"` for the covariance matrix calculated using the inverse
  of numerical Hessian matrix, and `"QMLE"` or `"ROBUST"` for the covari-
  ance matrix calculated using the asymptotic result for quasi-maximum
  likelihood estimation, sometimes referred to as the robust covariance
  matrix.

VALUE
  model type and summary statistics.

SEE ALSO
  `vcov.mgarch`, `summary` , `summary.mgarch` .

summary.forecast Summary Method for `forecast` Objects

DESCRIPTION

Returns an object of class `"summary.forecast"` for a `"forecast"` object. A null value will be returned if printing is invoked.

```
summary.forecast(object, ...)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"forecast"`.

VALUE

an object of class `"summary.forecast"` which is essentially the same as `object`. See `forecast.object` for details.

SEE ALSO

`forecast.object`.

summary.mgarch Summary for an Object of Fitted GARCH Object

DESCRIPTION

Summarizes an object of class inheriting `"mgarch"` .

summary.mgarch(x, max.lag=12, method="OP")

REQUIRED ARGUMENTS

x: an object of class inheriting `"mgarch"`, usually an object generated by the function `garch` or `mgarch`.

OPTIONAL ARGUMENTS

max.lag: maximum number of lags used in Lagrange Multiplier test and Ljung-Box test.

cov: a character string specifying the type of covariance matrix to be used. Valid choices are `"OP"` for outer product version of the covariance matrix, `"HESSIAN"` for the covariance matrix calculated using the inverse of numerical Hessian matrix, and `"QMLE"` or `"ROBUST"` for the covariance matrix calculated using the asymptotic result for quasi-maximum likelihood estimation, sometimes referred to as the robust covariance matrix.

VALUE

model type and summary statistics.

SEE ALSO

`vcov.mgarch`, `summary` , `summary.fgarch` .

summary.mimic Summary Method for `mimic` Objects

DESCRIPTION

Returns an object of class `"summary.mimic"` for a `"mimic"` object. A null value will be returned if printing is invoked.

```
summary.mimic(object, n.top=5)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"mimic"`.

OPTIONAL ARGUMENTS

`n.top` : an integer specifying the number of top positions to include in the summary.

VALUE

an object of class `"summary.mimic"` which summarizes the `n.top` top long and short positions in the factor mimicking portfolio.

SEE ALSO

`mimic`,`plot.summary.mimic` .

EXAMPLE

```
folio.mf = mfactor(folio.dat, 15)
summary(mimic(folio.mf))
```

summary.NLSUR Summary Method for a Nonlinear SUR Models

DESCRIPTION

> Returns some summary statistics for a nonlinear seemingly unrelated regressions model. A null value will be returned if printing is invoked.

> `summary.NLSUR(object)`

REQUIRED ARGUMENTS

`object:` an object inheriting from class `"NLSUR"`, usually returned by `NLSUR` function.

VALUE

> an object of class `"summary.NLSUR"`, which is the same as `object` with following additional components:

`coef:` a list of matrices, with each matrix having four columns, containing the coefficient estimates, their standard errors, the corresponding t statistics and p-values.

`rsq:` a numeric vector giving the multiple R-squared statistics for each equation in the model.

`arsq:` a numeric vector giving the adjusted R-squared statistics for each equation in the model.

`dw:` a numeric vector giving the Durbin-Watson test statistics for autocorrelation in the residuals.

SEE ALSO

> `summary`.

summary.OLS Summary Method for Ordinary Least Squares Models

DESCRIPTION

Returns some summary statistics for an ordinary least squares model. A null value will be returned if printing is invoked.

```
summary.OLS(object, correction=NULL, correlation=F,
            bandwidth=NULL, window="bartlett")
```

REQUIRED ARGUMENTS

`object`: an object inheriting from class `"OLS"`, usually returned by `OLS` function.

OPTIONAL ARGUMENTS

`correction`: a character string specifying the correction method for the covariance matrix. The default is `NULL` for no correction. Other choices are: `"white"` or `"hc"` for White's correction for heteroskedasticity, and `"nw"` or `"hac"` for Newey-West correction for heteroskedasticity and autocorrelation.

`correlation`: a logical flag: if `TRUE`, then the correlation matrix for the coefficients is included in the summary. The default is `FALSE`.

`bandwidth`: the bandwidth of the kernel window used to estimate the long run covariance matrix if `correction="nw"`, and ignored otherwise. By default, it is set to `floor(4*(N/100)^(2/9))` where `N` is the length of the response variable in the regression.

`window`: a character string specifying the kernel window used to estimate the long run covariance matrix if `correction="nw"`, and ignored otherwise. Currently only the following are supported: `"bartlett"` for a Bartlett or triangular kernel, and `"rectangular"` for a rectangular kernel.

VALUE

an object of class `"summary.OLS"`, which is the same as `object` with following additional components:

`coef`: a matrix with four columns, containing the coefficient estimates, their standard errors, the corresponding t statistics and p-values.

`residuals`: a vector of some summary statistics of the residuals from the fit, which includes: minimum, the first quartile, median, the third quartile, and maximum.

`df`: the number of degrees of freedom for the model and for residuals.

sigma: the residual standard error estimate.

rsq: the multiple R-squared statistic for the model.

arsq: the adjusted R-squared statistic for the model.

fstat: numeric vector of length three giving the F test for the regression. The first element is the statistic and the last two elements are the degrees of freedom.

dw: the Durbin-Watson test statistic for autocorrelation in the residuals.

tests: a 2 x 2 matrix giving the test statistics and P-values for Jarque-Bera normality test and Ljung-Box test of the residuals.

cov.unscaled: the unscaled covariance matrix; i.e, a matrix such that multiplying it by an estimate of the error variance produces an estimated covariance matrix for the coefficients.

correl: the computed correlation matrix for the coefficients in the model.

REFERENCES

Newey, W., and West, K. (1987). A simple positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55:703-708.

White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48:817-838.

SEE ALSO

vcov.OLS, summary .

EXAMPLE

summary(OLS(Oil~Market, data=oilcity))

summary.outliers Summary Method for `outliers` Objects

DESCRIPTION

Returns an object of class `"summary.outliers"` containing a summary list for an `"outliers"` object. A null value will be returned if printing is invoked.

```
summary.outliers(object)
```

REQUIRED ARGUMENTS

`object:` an object of class `"outliers"`.

VALUE

a list is returned with the following components:

`nout:` number of outliers (and level shifts) detected.

`outliers.table:` the index (or `"timeDate"` information if the argument `data` passed to `arima.rob` is of class `"timeSeries"`), type, impact and t value for each of the detected outlier (or level shift).

`sigma0:` the estimate of the innovation scale before correcting outliers.

`sigma:` the estimate of the innovation scale after correcting outliers.

DETAILS

This function is a method for the generic function `summary` for class `"outliers"`. It can be invoked by calling `summary` for an object of the appropriate class, or directly by calling `summary.outliers` regardless of the class of the object.

SEE ALSO

`outliers`,`outliers.object` ,`summary` .

EXAMPLE

```
frip.rr = arima.rob(log(frip.dat)~1, p=2, d=1)
frip.outliers = outliers(frip.rr)
summary(frip.outliers)
```

summary.rollOLS Summary Method for Rolling OLS Models

DESCRIPTION

Returns some summary statistics for a rolling OLS model. A null value will be returned if printing is invoked.

```
summary.rollOLS(object, ...)
```

REQUIRED ARGUMENTS

object: an object inheriting from class `"rollOLS"`, usually returned by `rollOLS` function.

VALUE

an object of class `"summary.rollOLS"`, which is the same as `object` with following additional components:

tstat: a matrix giving the t-statistics for coefficient estimates with each row from each rolling sample.

pval: a matrix giving the p-values for coefficient estimates with each row from each rolling sample.

SEE ALSO

`summary`.

summary.SEMIFAR Summary Method for SEMIFAR Models

DESCRIPTION

Returns some summary statistics for a semiparametric fractional AR model. A null value will be returned if printing is invoked.

```
summary.SEMIFAR(object)
```

REQUIRED ARGUMENTS

object: an object inheriting from class `"SEMIFAR"`, usually returned by `SEMIFAR` function.

VALUE

an object of class `"summary.SEMIFAR"`, which is the same as `object` with the following change:

coef: a matrix with four columns, containing the coefficient estimates, their standard errors, the corresponding t statistics and p-values.

SEE ALSO

`SEMIFAR.object`, `summary` .

summaryStats Data Summary

DESCRIPTION
>   Returns some useful summary statistics of a numeric object.

>   `summaryStats(x)`

REQUIRED ARGUMENTS

>   x: a numeric object. Missing values (`NA`) are removed before computation.

VALUE
>   an object of class `"summaryStats"`.

DETAILS
>   The class `"summaryStats"` is an S version 3 class, which has four components: `quantile`,`moments` , `n`, and `na`. The `quantile` component contains the following: minimum, first quartile, median, third quartile, and maximum. The`moments` component contains the following sample moments: mean, standard deviation, skewness, and kurtosis. The `n` and `na` components contain the total number of observations and number of missing values respectively.

SEE ALSO
>   `summary`.

EXAMPLE
>   `summaryStats(nelson.dat[,"WG"])`

summary.SUR Summary Method for Seemingly Unrelated Regressions
Models

DESCRIPTION

Returns some summary statistics for a seemingly unrelated regressions
model. A null value will be returned if printing is invoked.

```
summary.SUR(object)
```

REQUIRED ARGUMENTS

object: an object inheriting from class `"SUR"`, usually returned by `SUR` func-
tion.

VALUE

an object of class `"summary.SUR"`, which is the same as `object` with
following additional components:

coef: a list of matrices, with each matrix having four columns, containing
the coefficient estimates, their standard errors, the corresponding t
statistics and p-values.

rsq: a numeric vector giving the multiple R-squared statistics for each equa-
tion in the model.

arsq: a numeric vector giving the adjusted R-squared statistics for each
equation in the model.

dw: a numeric vector giving the Durbin-Watson test statistics for autocor-
relation in the residuals.

SEE ALSO

`summary`.

summary.unitroot Summary Method for Unit Root Test Objects

DESCRIPTION

Returns some summary statistics for a unit root regression. A null value will be returned if printing is invoked.

```
summary.unitroot(object, correlation=T)
```

REQUIRED ARGUMENTS

object: an object of class `"unitroot"`, usually returned by the `unitroot` function.

OPTIONAL ARGUMENTS

correlation: a logical flag: if `TRUE`, then the correlation matrix for the coefficients is included in the summary. The default is `TRUE`.

VALUE

an object with class `"summary.unitroot"`. The class `"summary.unitroot"` inherits from `"unitroot"` and a `"summary.unitroot"` object has the following additional components:

rsq: the multiple R-squared statistic for the unit root regression.

arsq: the adjusted R-squared statistic for the unit root regression.

fstat: numeric vector of length three giving the F test for the unit root regression. The first element is the statistic and the last two elements are the degrees of freedom.

dw: the Durbin-Watson test statistic for autocorrelation in the residuals.

correl: the computed correlation matrix for the coefficients in the regression.

SEE ALSO

unitroot.object, unitroot , summary .

EXAMPLE

```
wg.unit = unitroot(log(nelson.dat[,"WG"]), ar.order=5, na.rm=T)
summary(wg.unit)
```

summary.VAR Summary Method for VAR Models

DESCRIPTION

Returns an object of class `"summary.VAR"` for a VAR model. A null value will be returned if printing is invoked.

`summary.VAR(object, unbiased=T, info=T)`

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"VAR"`.

OPTIONAL ARGUMENTS

`unbiased` : a logical flag; if `TRUE`, unbiased estimate of error covariance will be used; if `FALSE`, the maximum likelihood estimate of error covariance will be used. The default is `TRUE`.

`info` : a logical flag; if `TRUE`, the information criteria will be computed. The default is `TRUE` for `"VAR"` objects and `FALSE` for `"BVAR"` objects.

VALUE

an object of class `"summary.VAR"` which is the same as `object` with following additional components:

`stderr` : the standard errors of the estimated coefficients.

`tStats` : the t-statistics of the estimated coefficients.

`rsq` : a matrix with three rows, with the first row representing the regression r-squared for each equation, the second row the adjusted r-squared for each equation, and the third row the residual scale estimate for each equation.

`info` : a logical flag, the same as the argument `info`.

SEE ALSO

`VAR.object`.

summary.VECM Summary Method for VECM Models

DESCRIPTION

Returns an object of class `"summary.VECM"` for a vector error correction model. A null value will be returned if printing is invoked.

```
summary.VECM(object, unbiased=T)
```

REQUIRED ARGUMENTS

`object` : an object inheriting from class `"VECM"`.

OPTIONAL ARGUMENTS

`unbiased` : a logical flag; if `TRUE`, unbiased estimate of error covariance will be used; if `FALSE`, the maximum likelihood estimate of error covariance will be used. The default is `TRUE`.

VALUE

an object of class `"summary.VECM"` which is the same as `object` with following additional components:

`stderr` : the standard errors of the estimated coefficients.

`tStats` : the t-statistics of the estimated coefficients.

`rsq` : a matrix with three rows, with the first row representing the regression r-squared for each equation, the second row the adjusted r-squared for each equation, and the third row the residual scale estimate for each equation.

SEE ALSO

`VECM.object`.

surex1.ts Exchange Rate Spot Returns and Forward Premium Data

SUMMARY

This is a monthly `"timeSeries"` object from March 1976 to June 1996, with the following twelve columns:

`USCN.FP.lag1`: one month forward premium between U.S. Dollar and Canadian Dollar.

`USCNS.diff`: future returns on spot rate between U.S. Dollar and Canadian Dollar.

`USDM.FP.lag1`: one month forward premium between U.S. Dollar and Deutsche Mark.

`USDMS.diff`: future returns on spot rate between U.S. Dollar and Deutsche Mark.

`USFR.FP.lag1`: one month forward premium between U.S. Dollar and French Franc.

`USFRS.diff`: future returns on spot rate between U.S. Dollar and French Franc.

`USIL.FP.lag1`: one month forward premium between U.S. Dollar and Italian Lira.

`USILS.diff`: future returns on spot rate between U.S. Dollar and Italian Lira.

`USJY.FP.lag1`: one month forward premium between U.S. Dollar and Japanese Yen.

`USJYS.diff`: future returns on spot rate between U.S. Dollar and Japanese Yen.

`USUK.FP.lag1`: one month forward premium between U.S. Dollar and British Pound.

`USUKS.diff`: future returns on spot rate between U.S. Dollar and British Pound.

SOURCE

Eric, Z. (2000). Cointegration and forward and spot exchange rate regressions. *Journal of International Money and Finance*, 19(6):785-812. 6:387-401.

SUR.fit Fitting of Seemingly Unrelated Regressions Models

DESCRIPTION

Called by the `SUR` function to fit the models. It is not supposed to be called by the users directly.

```
SUR.fit(X, Y, iterate=F, tol=1e-06, df=1, trace=T)
```

REQUIRED ARGUMENTS

X : a `list` object, with each component being the model matrix for each equation.

Y : a `list` object, with each component being the response for each equation.

OPTIONAL ARGUMENTS

iterate : a logical flag: if `TRUE`, the feasible GLS estimation is iterated till the parameters converge. The default is `FALSE`.

tol : a small number that determines the convergence of parameters when `iterate=T`. This is ignored if `iterate=F`. The default is set to `1e-6`.

df : an integer with value in the range `[1,4]`. This determines the degree of freedom to use in computing the initial residual covariance matrix. If `df=1`, the feasible GLS estimates are unbiased; if `df=2`, the feasible GLS estimates are minimum MSE estimates; if `df=3`, the sample size is used as the degree of freedom; if `df=4`, `sqrt((T-Li)*(T-Lj))` is used as the degree of freedom for the `(i,j)` element of the residual covariance matrix, where `T` is the sample size, `Li` is the number of parameters in `i`-th equation, and `Lj` is the number of parameters in `j`-th equation.

trace : a logical flag: if `TRUE`, a message will be printed on the screen when the GLS estimation is iterated. This is ignored if `iterate=F`. The default is `TRUE`.

SUR.object Seemingly Unrelated Regressions Model Objects

DESCRIPTION

These are objects of class `"SUR"` which represent the fit of a seemingly unrelated regressions model.

GENERATION

This class of objects is returned from the `SUR` function.

METHODS

The `"SUR"` class of objects currently has methods for the following generic functions:

`coef`, `residuals`, `fitted`, `vcov`, `plot`, `print`, `summary`.

STRUCTURE

The following components must be present in a legitimate `"SUR"` object:

`call` : an image of the call that produced the object.

`coef` : a numeric vector giving all the estimated coefficients.

`residuals` : a rectangular object with each column containing the residuals from each equation.

`fitted` : a rectangular object with each column containing the fitted values from each equation.

`cov` : the covariance matrix of all the estimated coefficients.

`Sigma` : a matrix giving the residual covariance estimate.

`X.k` : a list with length equal to the number of equations in the model, where each component of the list contains the number of coefficients in each equation.

`df` : the choice for degree of freedom as in the original call which generated the object.

`n.na` : the number of missing values in the original data set, if present.

SEE ALSO

SUR.

SUR Fit a Seemingly Unrelated Regressions Model

DESCRIPTION

Fits a seemingly unrelated regressions model to multivariate data.

```
SUR(formulas, data, subset, na.rm=F, start=NULL, end=NULL,
    contrasts=NULL, df=1, tol=1e-06, iterate=F, trace=T, ...)
```

REQUIRED ARGUMENTS

formulas : a list of `formula` objects, with each representing one equation in
the multivariate system.

data : a `data.frame` or a `"timeSeries"` object with a `data.frame` in the
data slot. This will be used to evaluate the variables specified in
`formulas`.

OPTIONAL ARGUMENTS

subset : an expression specifying which subset of observations should be used
in the fit. This can be a logical vector (which is replicated to have
length equal to the number of observations), a numeric vector indicat-
ing the observation numbers to be included, or a character vector of
the observation names that should be included. All observations are
included by default.

na.rm : a logical flag: if `TRUE`, missing values will be removed before fitting
the model. The default is `FALSE`.

start : a character string which can be passed to `timeDate` function to specify
the starting date for the estimation. This can only be used if the `data`
argument is a `"timeSeries"` data frame. The default is `NULL`.

end : a character string which can be passed to `timeDate` function to specify
the ending date for the estimation. This can only be used if the `data`
argument is a `"timeSeries"` data frame. The default is `NULL`.

contrasts : a list giving contrasts for some or all of the factors appearing in
the model formula. An element in the list should have the same name
as the factor variable it encodes, and it should be either a contrast
matrix (any full-rank matrix with as many rows as there are levels
in the factor), or a function that computes such a matrix given the
number of levels.

df : an integer with value in the range [1,4]. This determines the degree
of freedom to use in computing the initial residual covariance matrix.
If `df=1`, the feasible GLS estimates are unbiased; if `df=2`, the feasible

GLS estimates are minimum MSE estimates; if `df=3`, the sample size is used as the degree of freedom; if `df=4`, `sqrt((T-Li)*(T-Lj))` is used as the degree of freedom for the `(i,j)` element of the residual covariance matrix, where `T` is the sample size, `Li` is the number of parameters in `i`-th equation, and `Lj` is the number of parameters in `j`-th equation.

`tol` : a small number that determines the convergence of parameters when `iterate=T`. This is ignored if `iterate=F`. The default is set to `1e-6`.

`iterate` : a logical flag: if `TRUE`, the feasible GLS estimation is iterated till the parameters converge. The default is `FALSE`.

`trace` : a logical flag: if `TRUE`, a message will be printed on the screen when the GLS estimation is iterated. This is ignored if `iterate=F`. The default is `TRUE`.

`...` : any optional arguments that can be passed to the `timeDate` function to parse the `start` or `end` specification.

VALUE

an object of class `"SUR"` which represents the fit of the seemingly unrelated regressions model. See `SUR.object` for details.

DETAILS

Constrained SUR estimation can be carried by using `NLSUR` function. See the help file for `NLSUR` for details.

REFERENCES

Srivastava, V. K., and Dwivedi, T. D. (1979). Estimation of seemingly unrelated regression equations: a brief survey. *Journal of Econometrics*, 10:15-32.

SEE ALSO

NLSUR, SUR.object .

EXAMPLE

```
# simulate a multivariate data set.
set.seed(10)
x1 <- abs(rnorm(100))
x2 <- x1^2
sim.dat <- data.frame(x1=x1,
                      x2=x2,
                      y1=1+2*x1+rnorm(100),
                      y2=4+5*x2+rnorm(100))
# estimate an SUR model
SUR(list(y1~x1, y2~x2), data=sim.dat)
```

TA.accel Technical Indicator: Acceleration

DESCRIPTION

>Returns the technical indicator acceleration given a time series of security price.

>   TA.accel(x, n=12, trim=T)

REQUIRED ARGUMENTS

>x : a vector, or a `"timeSeries"` object.

OPTIONAL ARGUMENTS

>n : a positive integer specifying the number of periods to be used for calculating momentum. The default uses a 12-term momentum.

>trim : a logical flag: if `TRUE`, the first `n-1` missing values in the output are trimmed; if `FALSE`, they are saved in the returned value.

VALUE

>a vector or `"timeSeries"` object representing the acceleration of a security price, i.e., the difference between the current momentum and the previous momentum.

REFERENCES

>Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

>Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

>`TA.momentum`,`TA.roc` .

TA.adi Technical Indicator: Accumulation/Distribution Indicator

DESCRIPTION

Returns the technical indicator accumulation/distribution given the time series of high, low and close security prices along with the trading volume.

```
TA.adi(high, low, close, volume)
```

REQUIRED ARGUMENTS

high : a vector, or a `"timeSeries"` object representing the highest price of a security for each period.

low : a vector, or a `"timeSeries"` object representing the lowest price of a security for each period.

close : a vector, or a `"timeSeries"` object representing the closing price of a security for each period.

volume : a vector, or a `"timeSeries"` object representing the trading volume of a security for each period.

VALUE

a vector or a `"timeSeries"` object representing the A/D indicator.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`TA.adoscillator`,`TA.chaikino`,`TA.nvi`,`TA.pvi`,`TA.obv`,`TA.pvtrend`
.

TA.adoscillator Technical Indicator: Accumulation/Distribution Oscillator

DESCRIPTION

> Returns the technical indicator accumulation/distribution oscillator given the time series of open, high, low and close security prices.


```
TA.adoscillator(open, high, low, close)
```

REQUIRED ARGUMENTS

  open : a vector, or a `"timeSeries"` object representing the opening price of a security for each period.

  high : a vector, or a `"timeSeries"` object representing the highest price of a security for each period.

   low : a vector, or a `"timeSeries"` object representing the lowest price of a security for each period.

 close : a vector, or a `"timeSeries"` object representing the closing price of a security for each period.

 VALUE

> a vector or a `"timeSeries"` object representing the A/D oscillator.

REFERENCES

> Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

> Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

> `TA.adi`,`TA.chaikinv`,`TA.garmanKlass` .

TA.bollinger   Technical Indicator: Bollinger Band

DESCRIPTION

   Returns the technical indicator Bollinger band given a time series of
   security prices.

   ```
   TA.bollinger(x, n=20, n.sd=2, trim=T, na.rm=F)
   ```

REQUIRED ARGUMENTS

   `x` : a vector, or a `"timeSeries"` object.

OPTIONAL ARGUMENTS

   `n` : a positive integer specifying the number of periods to be used for
       calculating SMA. The default uses a 20-term SMA.

   `n.sd` : a positive number specifying the width of the bands in terms of stan-
       dard deviations. The default is to compute the bands as two standard
       deviations above and below a 20-term SMA.

   `trim` : a logical flag: if `TRUE`, the first `n-1` missing values in the output are
       trimmed; if `FALSE`, they are saved in the returned value.

   `na.rm` : a logical flag: if `TRUE`, missing values will be removed before computing
       Bollinger band. The default is `FALSE`.

VALUE

   a matrix or a `"timeSeries"` object with three columns, the first being
   the upper band, the second the moving average, and the third the lower
   band.

REFERENCES

   Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub
   Co.

   Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Tech-
   nical Market Indicators*. McGraw-Hill.

SEE ALSO

   `SMA`,`rollVar` .

TA.chaikino Technical Indicator: Chaikin's Oscillator

DESCRIPTION

Returns the technical indicator Chaikin's oscillator given the time series of high, low and close security prices along with the trading volume.

```
TA.chaikino(high, low, close, volume, n.long=10, n.short=3,
            start="average", na.rm = F)
```

REQUIRED ARGUMENTS

high : a vector, or a `"timeSeries"` object representing the highest price of a security for each period.

low : a vector, or a `"timeSeries"` object representing the lowest price of a security for each period.

close : a vector, or a `"timeSeries"` object representing the closing price of a security for each period.

volume : a vector, or a `"timeSeries"` object representing the trading volume of a security for each period.

OPTIONAL ARGUMENTS

n.long : a positive integer specifying the number of periods for the longer window. The default uses a 10-term window.

n.short : a positive integer specifying the number of periods for the shorter window. The default uses a 3-term window.

start : a character string or a number that specifies the first number to use to start the EWMA filter. If `start` is a character string, the valid choices are: `"average"`, and `"first"`. When `start="average"`, the average of the first `n` observations is used as the starting value; when `start="first"`, the first observation in `x` is used as the starting value. If `start` is a number, then it will be used as the starting value. The default is `"average"`.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computing EWMA. The default is `FALSE`.

VALUE

a vector or a `"timeSeries"` object representing the Chaikin's oscillator.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`TA.adi`,`TA.chaikinv` ,`TA.nvi` ,`TA.pvi` ,`TA.pvtrend` ,`TA.obv` .

TA.chaikinv   Technical Indicator: Chaikin's Volatility

DESCRIPTION

Returns the technical indicator Chaikin's volatility given the time series of high and low security prices.

```
TA.chaikinv(high, low, n.range=10, n.change=10, trim=T,
    start="average", na.rm=F)
```

REQUIRED ARGUMENTS

high : a vector, or a `"timeSeries"` object representing the highest price of a security for each period.

low : a vector, or a `"timeSeries"` object representing the lowest price of a security for each period.

OPTIONAL ARGUMENTS

n.range : a positive number specifying the number of periods to use for computing the EWMA of trading ranges. The default uses a 10-term EWMA.

n.change : a positive number specifying the number of periods to use for computing the percentage changes. The default uses 10 periods.

trim : a logical flag: if `TRUE`, the missing values at the beginning of the returned object will be removed; otherwise, they will be kept in the returned object.

start : a character string or a number that specifies the first number to use to start the EWMA filter. If `start` is a character string, the valid choices are: `"average"`, and `"first"`. When `start="average"`, the average of the first `n` observations is used as the starting value; when `start="first"`, the first observation in `x` is used as the starting value. If `start` is a number, then it will be used as the starting value. The default is `"average"`.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computing EWMA. The default is `FALSE`.

VALUE

a vector or a `"timeSeries"` object representing Chaikin's volatility.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`TA.adoscillator,TA.chaikino ,TA.garmanKlass .`

TA.garmanKlass Technical Indicator: Garman-Klass Volatility

DESCRIPTION

Returns the Garman-Klass estimate of volatility given the time series of open, high, low and close security prices.

```
TA.garmanKlass(open, high, low, close)
```

REQUIRED ARGUMENTS

open : a vector, or a "timeSeries" object representing the opening price of a security for each period.

high : a vector, or a "timeSeries" object representing the highest price of a security for each period.

low : a vector, or a "timeSeries" object representing the lowest price of a security for each period.

close : a vector, or a "timeSeries" object representing the closing price of a security for each period.

VALUE

a vector or a "timeSeries" object representing the Garman-Klass volatility.

REFERENCES

Garman, M. B., and Klass, M. J. (1980). On the estimation of security price volatility from historical data. *Journal of Business*, 53(1):67-78.

SEE ALSO

TA.adoscillator, TA.chaikinv .

tail.index Measure of Tail Dependence

DESCRIPTION

Computes tail index (a measure of tail dependence) for a parametric copula or an empirical copula.

```
tail.index(copula, ...)
```

REQUIRED ARGUMENTS

`copula`: an object of class `"copula"`.

`...`: In case if copula is an empirical copula, any graphical parameters.

VALUE

Tail index for upper and lower tails for a parametric copula. For an empirical copula a plot of tail index estimate.

DETAILS

This is a generic function. Almost all copula classes (including empirical copula) have their method functions. A plot of tail index estimate is generated on a graphical device if input is of `empirical.copula` class.

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

SEE ALSO

`empirical.copula.object`, `Kendalls.tau` .

EXAMPLE

```
ec <- empirical.copula(rcopula(bb5.copula(1.5,0.7),1000))
tail.index(ec)
tail.index(bb5.copula(1.5,0.7))
```

tailplot Plot Tail Estimate From GPD Model

DESCRIPTION

Interacts with the output of `gpd` function to produce a plot of the tail of the underlying distribution of the data. This is Option 2 of `plot.gpd`, but `tailplot` enables the user to bypass the menu of the former.

```
plot.gpd(gpd.obj, optlog=NA, extend=1.5, labels=T, ...)
```

REQUIRED ARGUMENTS

`gpd.obj`: a "gpd" object. This is usually returned by the `gpd` function.

OPTIONAL ARGUMENTS

`optlog`: a character string giving a particular choice of logarithmic axes (for plots 1 and 2): "x" for x-axis only; "y" for y-axis only; "xy" for both axes; "" for neither axis. The default is `NA` which generates logarithmic x-axis for plot 1 and both axes on logarithmic scale for plot 2.

`extend`: a numerical value greater than 1 specifying how far x-axis should extend as a multiple of the largest data value (for plots 1 and 2 only). This argument is useful for showing estimated quantiles beyond data. The default is `1.5`.

`labels`: a logical flag for plots 1 and 2 specifying whether or not axes should be labeled. The default is `TRUE`.

tailplot causes the creation of an object `lastcurve` in frame 0 containing details of the plot. This is needed if quantile estimates are to be added.

SEE ALSO

gpd, `plot.gpd` , `shape` , `quant` .

EXAMPLE

```
out = gpd(danish,10)
tailplot(out)
```

TA.macd   Technical Indicator: MACD

DESCRIPTION

Returns the technical indicator Moving Average Convergence/Divergence (MACD) given a time series of security price.

```
TA.macd(x, n.short=12, n.long=26, n.signal=9, start="average",
    na.rm=F)
```

REQUIRED ARGUMENTS

`x` : a vector, or a `"timeSeries"` object.

OPTIONAL ARGUMENTS

`n.short` : a positive integer specifying the number of periods to be used for calculating the short window EWMA. The default uses a 12-term EWMA.

`n.long` : a positive integer specifying the number of periods to be used for calculating the long window EWMA. The default uses a 26-term EWMA.

`n.signal` : a positive integer specifying the number of periods to be used for calculating the signal line. The default uses a 9-term EWMA.

`start` : a character string or a number that specifies the first number to use to start the EWMA filter. If `start` is a character string, the valid choices are: `"average"`, and `"first"`. When `start="average"`, the average of the first `n` observations is used as the starting value; when `start="first"`, the first observation in `x` is used as the starting value. If `start` is a number, then it will be used as the starting value. The default is `"average"`.

`na.rm` : a logical flag: if `TRUE`, missing values will be removed before computing EWMA. The default is `FALSE`.

VALUE

a matrix or a `"timeSeries"` object with two columns, the first representing MACD, and the second the signal line.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`EWMA`,`TA.stochastic` .

TA.medprice   Technical Indicator: Median Price

DESCRIPTION

>   Returns the technical indicator median price given the time series of
>   high and low security prices.

>   `TA.medprice(high, low)`

REQUIRED ARGUMENTS

  `high` : a vector, or a `"timeSeries"` object representing the highest price of
  a security for each period.

  `low` : a vector, or a `"timeSeries"` object representing the lowest price of a
  security for each period.

 VALUE

>   a vector or `"timeSeries"` object representing the median price, which
>   are simply the averages of the high and low prices in each period.

REFERENCES

>   Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub
>   Co.

>   Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

>   `TA.typicalPrice`,`TA.wclose` .

TA.momentum Technical Indicator: Momentum

DESCRIPTION

      Returns the technical indicator momentum given a time series of security price.

      `TA.momentum(x, n=12, trim=T)`

REQUIRED ARGUMENTS

    `x` : a vector, or a `"timeSeries"` object.

OPTIONAL ARGUMENTS

    `n` : a positive integer specifying the number of periods to be used for calculating momentum. The default computes a 12-term momentum.

  `trim` : a logical flag: if `TRUE`, the first `n-1` missing values in the output are trimmed; if `FALSE`, they are saved in the returned value.

 VALUE

      a vector or `"timeSeries"` object which measures the amount that the security price has changed in a given time period.

REFERENCES

      Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

      Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

      `TA.accel`,`TA.roc` ,`TA.rsi` .

TA.obv Technical Indicator: On Balance Volume Indicator

DESCRIPTION

> Returns the technical indicator on balance volume given the time series of closing security price and trading volume.

> `TA.obv(close, volume)`

REQUIRED ARGUMENTS

`close` : a vector, or a `"timeSeries"` object representing the closing price of a security for each period.

`volume` : a vector, or a `"timeSeries"` object representing the trading volume of a security for each period.

VALUE

> a vector or `"timeSeries"` object representing the on balance volume indicator.

REFERENCES

> Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

> Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

> `TA.adi`,`TA.chaikino` ,`TA.nvi` ,`TA.pvi` ,`TA.pvtrend` .

TA.pvi Technical Indicator: Positive/Negative Volume Index

DESCRIPTION

Returns the technical indicator positive/negative volume index given the time series of closing security price and trading volume.

```
TA.psi(close, volume)
TA.nsi(close, volume)
```

REQUIRED ARGUMENTS

`close` : a vector, or a `"timeSeries"` object representing the closing price of a security for each period.

`volume` : a vector, or a `"timeSeries"` object representing the trading volume of a security for each period.

VALUE

a vector or `"timeSeries"` object representing the positive volume index or negative volume index.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`TA.adi`,`TA.chaikino` ,`TA.obv` ,`TA.pvtrend` .

TA.pvtrend Technical Indicator: Price-Volume Trend

DESCRIPTION

Returns the technical indicator price-volume trend given the time series of closing security price and trading volume.

```
TA.pvtrend(close, volume)
```

REQUIRED ARGUMENTS

`close` : a vector, or a `"timeSeries"` object representing the closing price of a security for each period.

`volume` : a vector, or a `"timeSeries"` object representing the trading volume of a security for each period.

VALUE

a vector or `"timeSeries"` object representing the price-volume trend.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`TA.adi`,`TA.chaikino` ,`TA.nvi` ,`TA.pvi` ,`TA.obv` .

TA.roc Technical Indicator: Rate of Change of Price/Volume

DESCRIPTION

Returns the technical indicator rate of change given a time series of security price or trading volume.

```
TA.roc(x, n=12, trim=T)
```

REQUIRED ARGUMENTS

`x` : a vector, or a `"timeSeries"` object.

OPTIONAL ARGUMENTS

`n` : a positive integer specifying the number of periods to be used for calculating change. The default uses 12 periods.

`trim` : a logical flag: if `TRUE`, the first `n-1` missing values in the output are trimmed; if `FALSE`, they are saved in the returned value.

VALUE

a vector or `"timeSeries"` object representing the rate of change.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`TA.accel`,`TA.momentum` .

TA.rsi Technical Indicator: Relative Strength Index

DESCRIPTION

Returns the technical indicator Relative Strength Index (RSI) given a time series of security price.

```
TA.rsi(x, n=14, simple=T, trim=T, start="average", na.rm=F)
```

REQUIRED ARGUMENTS

x : a vector, or a `"timeSeries"` object.

OPTIONAL ARGUMENTS

n : a positive integer specifying the number of periods to be used for calculating RSI. The default uses 14 periods.

simple : a logical flag: if `TRUE`, a simple moving average is used; if `FALSE`, an exponential weighted moving average is used. The default is `TRUE`.

trim : a logical flag. It is only used if `simple=T`. See `SMA` for details. The default is `TRUE`.

start : a character string or a number that specifies the first number to use to start the EWMA filter. If `start` is a character string, the valid choices are: `"average"`, and `"first"`. When `start="average"`, the average of the first `n` observations is used as the starting value; when `start="first"`, the first observation in x is used as the starting value. If `start` is a number, then it will be used as the starting value. This is only used if `simple=F`. The default is `"average"`.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computing EWMA. The default is `FALSE`.

VALUE

a vector or a `"timeSeries"` object representing the Relative Strength Index.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`EWMA`,`SMA` .

TA.stochastic   Technical Indicator: Stochastic Oscillator

DESCRIPTION

Returns the technical indicator stochastic oscillator given the time series of high, low and close security prices.

```
TA.stochastic(high, low, close, n.k=10, n.d=3, type="slow",
    trim=T, na.rm=F)
```

REQUIRED ARGUMENTS

high : a vector, or a `"timeSeries"` object representing the highest price of a security for each period.

low : a vector, or a `"timeSeries"` object representing the lowest price of a security for each period.

close : a vector, or a `"timeSeries"` object representing the closing price of a security for each period.

OPTIONAL ARGUMENTS

n.k : a positive integer specifying the number of periods to use for computing the trading ranges. The default uses 10 periods.

n.d : a positive integer specifying the number of periods to use for computing the moving averages. The default uses 3 periods.

type : a character string specifying the type of stochastic oscillator to be computed. If `type="slow"`, slow stochastic will be computed; if `type="fast"`, fast stochastic will be computed. The default is `"slow"`.

trim : a logical flag: if `TRUE`, the missing values at the beginning of the output are trimmed; if `FALSE`, they are saved in the returned value.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computation. The default is `FALSE`.

VALUE

a matrix or `"timeSeries"` object representing the stochastic oscillator, with the first column being the %K line, and second the %D line.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`TA.williamsad`,`TA.williamsr` .

TA.typicalPrice Technical Indicator: Typical Price

DESCRIPTION

Returns the technical indicator typical price given the time series of high, low and close security prices.

```
TA.typicalPrice(high, low, close)
```

REQUIRED ARGUMENTS

high : a vector, or a `"timeSeries"` object representing the highest price of a security for each period.

low : a vector, or a `"timeSeries"` object representing the lowest price of a security for each period.

close : a vector, or a `"timeSeries"` object representing the closing price of a security for each period.

VALUE

a vector or `"timeSeries"` object representing the typical prices, which are simply the averages of the high, low, and close prices in each period.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`TA.medprice`, `TA.wclose` .

TA.wclose   Technical Indicator: Weighted Close

DESCRIPTION

Returns the technical indicator weighted close given the time series of high, low and close security prices.

`TA.wclose(high, low, close)`

REQUIRED ARGUMENTS

`high` : a vector, or a `"timeSeries"` object representing the highest price of a security for each period.

`low` : a vector, or a `"timeSeries"` object representing the lowest price of a security for each period.

`close` : a vector, or a `"timeSeries"` object representing the closing price of a security for each period.

VALUE

a vector or `"timeSeries"` object representing the weighted closes. The weighted close is very similar to typical price, except that it gives extra weight to the closing price.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`TA.medprice`,`TA.typicalPrice` .

TA.williamsad Technical Indicator: Williams' Accumulation/Distribution

DESCRIPTION

Returns the technical indicator Willimas' accumulation/distribution given the time series of high, low and close security prices.

```
TA.williamsad(high, low, close)
```

REQUIRED ARGUMENTS

high : a vector, or a `"timeSeries"` object representing the highest price of a security for each period.

low : a vector, or a `"timeSeries"` object representing the lowest price of a security for each period.

close : a vector, or a `"timeSeries"` object representing the closing price of a security for each period.

VALUE

a vector or a `"timeSeries"` object representing Williams A/D index.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

`TA.adi`,`TA.stochastic`,`TA.williamsr`.

TA.williamsr Technical Indicator: Williams' %R

DESCRIPTION

Returns the technical indicator Willimas' %R given the time series of high, low and close security prices.

```
TA.williamsr(high, low, close, n=20, trim=T, na.rm=F)
```

REQUIRED ARGUMENTS

high : a vector, or a "timeSeries" object representing the highest price of a security for each period.

low : a vector, or a "timeSeries" object representing the lowest price of a security for each period.

close : a vector, or a "timeSeries" object representing the closing price of a security for each period.

OPTIONAL ARGUMENTS

n : a positive integer specifying the number of periods to be used for calculating the highest high and lowest low. The default uses 20 periods.

trim : a logical flag: if TRUE, the first n-1 missing values in the output are trimmed; if FALSE, they are saved in the returned value.

na.rm : a logical flag: if TRUE, missing values will be removed before computation. The default is FALSE.

VALUE

a vector or a "timeSeries" object representing Williams' %R index.

REFERENCES

Achelis, S. B. (1995). *Technical Analysis from A to Z*. Probus Pub Co.

Colby, R. W., and Meyers, T. A. (1988). *The Encyclopedia of Technical Market Indicators*. McGraw-Hill.

SEE ALSO

TA.adi,TA.stochastic ,TA.williamsad .

tawn.copula.object   Tawn Copula Class Object

DESCRIPTION

These are S version 4 objects of class `"tawn.copula"` , which inherits from `copula` and `ev.copula` classes.

GENERATION

This class of objects is constructed from the `tawn.copula(alpha, beta, r)` function or `ev.copula` function.

METHODS

The `"tawn.copula"` class of objects currently has methods for the following generic functions:

`Afunc, AfirstDer, AsecondDerPhiDer, Hderiv, pcopula, dcopula, rcopula, contour.plot, tail.index, Kendalls.tau, Spearmans.rho, dcdx`.

Furthermore, following functions are implemented for this class: `persp.dcopula, persp.pcopula, contour.dcopula, contour.pcopula`.

STRUCTURE

The following slots must be present in a legitimate `"tawn.copula"` class object:

`parameters` : a vector of values for the parameter(s) `alpha` (between 0 and 1), `beta` (between 0 and 1), and `r` (greater or equal to 1).

`param.names` : `alpha, beta`, and `r`.

`param.lowbnd` : a vector with the same length as the `parameters` vector, containing the values of the lower bounds for the parameter(s). These values are used by the `fit.copula` function.

`param.upbnd` : similar to `param.lowbnd`, only values of the upper bounds for the parameter(s).

`message` : a message specifying the name of the parametric copula family (Tawn copula family), and the copula class from which it inherits (Extreme Value Copula).

REFERENCES

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

Tawn, J. A. (1988). Bivariate extreme value theory: models and estimation. *Biometrika*, 75:397-415.

SEE ALSO

`copula.object`,`ev.copula.object` ,`ev.copula` .

term.struct Fitting Term Structure of Interest Rates

DESCRIPTION

Fits the term structure of interest rates using quadratic spline, cubic spline, smoothing spline, Nelson-Siegel function, or Nelson-Siegel-Svensson function.

```
term.struct(rate, maturity, method="cubic", na.rm=F,
            input.type="yield", plot=T, k=NULL, cv=F,
            compounding.frequency=0, penalty=2, spar=0,
            ...)
```

REQUIRED ARGUMENTS

`rate`: a numeric vector giving the interest rates.

`maturity`: a numeric vector giving the maturities corresponding to the interest rates in `rate`. This is in units of years.

OPTIONAL ARGUMENTS

`method`: a character string specifying the method to use for fitting the term structure. Valid choices are: `"quadratic"` for quadratic spline, `"cubic"` for cubic spline, `"smooth"` for smoothing spline, `"ns"` for Nelson-Siegel function, and `"nss"` for Nelson-Siegel-Svensson function. The default is `"cubic"`.

`input.type`: a character string specifying the type of `rate`. Valid choices are as follows: `"spot"`, `"forward"`, and `"discount"`. This is needed if a rate conversion is necessary. See Details section. The default is `"spot"`.

`na.rm`: a logical flag: if `TRUE`, missing values will be removed before fitting the model. The default is `FALSE`.

`plot`: a logical flag: if `TRUE`, a plot of the fitted term structure will be shown. The default is `TRUE`.

`compounding.frequency`: an integer specifying the compounding frequency in a year. If `compounding.frequency=0`, then continuous compounding is used. This is needed if a rate conversion is necessary. See Details section. The default is 0.

`k`: an integer specifying the number of knot points. This is only used for quadratic or cubic splines. If `k=NULL`, it is set to `sqrt(n)` for quadratic spline, and `max(sqrt(n), 3)` for cubic spline.

`cv`: a logical flag: if `TRUE`, the ordinary cross validation score is computed; if `FALSE`, generalized cross validation score is computed. This is only used by smoothing spline method. The default is `FALSE`.

penalty: the penalty per degree of freedom in generalized cross validation. This is only used by smoothing spline method. The default is 2.

spar: the coefficient of the integrated second squared derivative penalty function. This is only used by smoothing spline method. The default is 0.

...: any optional argument that may be passed down to the plot function.

VALUE

an object of class `"term.struct"`. If the smoothing spline method is used, then the object inherits from `"smooth.spline"` class; otherwise, it inherits from `"lm"` class. A plot of the term structure will be shown in a graphical device if `plot=T`.

DETAILS

The Nelson-Siegel and Svensson functions and smoothing spline method apply to the spot interest rates, so a rate conversion will be necessary if the input is not a spot interest rate. The quadratic and cubic spline methods apply to the discount rate, so a rate conversion will be necessary if the input is not a discount rate.

REFERENCES

Fisher, M., Nychka, D., and Zervos, D. (1995). Fitting the term structure of interest rates with smoothing splines. Finance and Economics Discussion Series #1995-1, Board of Governors of the Federal Reserve System.

McCulloch, J. H. (1973). Measuring the term structure of interest rates. *Journal of Business*, 44:19-31.

McCulloch, J. H. (1975). The tax-adjusted yield curve. *Journal of Finance*, 30(3):811-830.

Nelson, C. R., and Siegel, A. F. (1987). Parsimonious modeling of yield curves. *Journal of Business*, 60(4):473-489.

Svensson, L. E. O. (1994). Estimating and interpreting forward interest rates: Sweden 1992-1994. NBER Working Paper No. 4871.

SEE ALSO

lm.object,smooth.spline .

EXAMPLE

```
term.struct(mk.zero2[21,], mk.maturity, na.rm=T, method="ns",
          input="spot", plot=T)
```

tslag   Time Series Lag/Lead Function

DESCRIPTION

Returns a lagged/led vector or matrix given a time series data.

```
tslag(x, k=1, trim=F)
```

REQUIRED ARGUMENTS

x : a vector, a matrix, or a `"timeSeries"` object. Missing values (`NA`) are allowed.

OPTIONAL ARGUMENTS

k : the number of positions the new series is to lag or lead the input series, with a positive value resulting in a lagged series and a negative value resulting in a leading series.

trim : a logical flag: if `TRUE`, the missing values at the beginning or end of the returned series will be trimmed. The default is `FALSE`.

VALUE

a lagged or leading time series of the original data.

SEE ALSO

`lag`, `shift` .

EXAMPLE

```
# create a lagged matrix
tslag(policy.dat[1:10,"GDP"], k=1:3)

# create a leading matrix
tslag(policy.dat[1:10,"GDP"], k=-(1:3))
```

unitroot.object Unit Root Regression Object

DESCRIPTION
> These are objects of class `"unitroot"`. They represent the fit of a unit root regression.

GENERATION
> This class of objects is returned from the `unitroot` function.

METHODS
> The `"unitroot"` class of objects currently has methods for the following generic functions:
>
> `coef`, `residuals`, `fitted`, `vcov`, `plot`, `print`, `summary`.

STRUCTURE
> The following components must be present in a legitimate `"unitroot"` object:

`coef:` the estimated coefficients of the unit root regression.

`residuals:` the residuals from the unit root regression.

`fitted:` the fitted values from the unit root regression.

`cov.unscaled:` the unscaled covariance matrix; i.e, a matrix such that multiplying it by an estimate of the error variance produces an estimated covariance matrix for the coefficients.

`sigma:` the residual standard error estimate for the unit root regression.

`sval:` the value of the unit root test statistic.

`reg:` a character string describing the method of the unit root test.

`bandwidth:` the bandwidth of the kernel used to estimate the long run variance if `type="pp"` .

`asymptotic:` a logical flag: if `TRUE`, the p-value of the test will be computed based on asymptotic results.

`na.message:` a character string describing the action taken if there are missing values in the data.

SEE ALSO
> `unitroot`.

unitroot Unit Root Tests

DESCRIPTION

Returns an object representing a test for unit root of a vector or time series.

```
unitroot(x, trend="c", method="adf", statistic="t",
         lags=1, bandwidth=NULL, window="bartlett",
         asymptotic=F, na.rm=F)
```

REQUIRED ARGUMENTS

x : a numeric vector or `"timeSeries"` vector.

OPTIONAL ARGUMENTS

trend : a character string describing the type of the unit root regression. Valid choices are: `"nc"` for a regression with no intercept (constant) nor time trend, and `"c"` for a regression with an intercept (constant) but no time trend, `"ct"` for a regression with an intercept (constant) and a time trend. The default is `"c"`.

method : a character string specifying the method of the test. Valid choices are: `"adf"` for augmented Dickey-Fuller test, and `"pp"` for Phillips-Perron test. The default is `"adf"`.

statistic : a character string specifying the test statistic. Valid choices are: `"t"` for t-statistic, and `"n"` for normalized statistic (sometimes referred to as the rho-statistic). The default is `"t"`.

lags : the autoregressive order to use in the unit root regression if `method="adf"`. If `method="pp"`, this is always set to one. The default is 1.

bandwidth : the bandwidth of the kernel used to estimate the long run variance if `method="pp"`. By default, it is set to `floor(4 * (N/100)^0.25)` where `N` is the length of the response variable in the unit root regression.

window : a character string specifying the kernel used to estimate the long run variance if `method="pp"`. Currently only the following are supported: `"bartlett"` for a Bartlett or triangular kernel, and `"rectangular"` for a rectangular kernel. The default is `"bartlett"`.

asymptotic : a logical flag: if `TRUE`, the p-value of the test is computed based on asymptotic results. The default is `FALSE`. The default is `FALSE`.

na.rm : a logical flag: if `TRUE`, missing values will be removed before computing the test statistic. The default is `FALSE`.

VALUE

an object of class `"unitroot"` representing the unit root test. See `unitroot.object`

for details.

REFERENCES

Dickey, D. A., and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427-431.

MacKinnon, J. G. (1996). Numerical distribution functions for unit root and cointegration tests. *Journal of Applied Econometrics*, 11:601-618.

Phillips, P. C. B., and Perron, P. (1988). Testing for a unit root in time series regression. *Biometrika*, 75(2):335-346.

SEE ALSO

`punitroot,qunitroot ,unitroot.object` .

EXAMPLE

```
unitroot(log(nelson.dat[,"WG"]), trend="c", stat="t",
         lags=5, na.rm=T)
```

uti.elec Electricity and Gas Prices and Their Log Returns

SUMMARY

This is a `"timeSeries"` object of 2 columns data: utility price and its log return on each of the 2125 trading days between 1/1/1992 to 5/29/2000.

VAR.ar2ma Moving Average Representation of a VAR Model

DESCRIPTION

Returns the moving average representation of a vector autoregression model, given a fitted `"VAR"` object. This is called to compute the impulse response function.

```
VAR.ar2ma(x, period=NULL)
```

REQUIRED ARGUMENTS

x : an object of class `"VAR"`.

OPTIONAL ARGUMENTS

period : the order of the moving average representation. By default, we set `period=p*(k-1) + 1`, where `p` is the order of the autoregressive model, and `k` is the dimension of the multivariate response.

VALUE

a three dimensional array of the moving average representation with dimension `c(k,k,period)`, where `k` is the number of variables in the multivariate response.

REFERENCES

Lutkepohl, H. (1990). *Introduction to Multiple Time Series Analysis*, Springer-Verlag.

SEE ALSO

`VECM2VAR`,`FARIMA.d2ar` , `arma2ma` .

EXAMPLE

```
ccoas.mat = as.matrix(ccoas.dat)
ccoas.mod = VAR(ccoas.mat~ar(13))
VAR.ar2ma(ccoas.mod, period=3)
```

varex.ts Real Stock Returns and Output Growth Data

SUMMARY

This is a monthly `"timeSeries"` object from July 1926 to December 2000, with the following four columns:

`MARKET.REAL`: continuously compounded real returns on S & P 500 index.

`RF.REAL`: real interest rate of 30-day U.S. Treasury bills.

`INF`: continuously compounded growth rate of U.S. CPI.

`IPG`: continuously compounded growth rate of U.S. industrial production.

VAR.fit Fitting of VAR Models

DESCRIPTION

Called by `VAR` to fit the models. It is not supposed to be called by the users directly.

```
VAR.fit(X, Y)
```

REQUIRED ARGUMENTS

X, Y : numeric matrices representing the regressors and multivariate response respectively.

VAR.formula Fit a Vector Autoregression Model Using a Formula

DESCRIPTION

Returns an object of class `"VAR"` that represents the fit of a VAR model, using a formula specification.

```
VAR.formula(formula, data, subset, na.rm=F, contrasts=NULL, start=NULL,
    end=NULL, demean="none", ...)
```

REQUIRED ARGUMENTS

formula : a `formula` object, with the response on the left of a $\sim$ operator and the terms, separated by + operators, on the right. The response must be a `"matrix"` object if the `data` argument is not specified.

OPTIONAL ARGUMENTS

data : a data frame or `"timeSeries"` data frame in which to interpret the variables named in the `formula` and `subset` arguments. If `data` is missing, the variables in the model formula should be in the search path, and cannot be `"timeSeries"` objects.

subset : an expression specifying which subset of observations should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), a numeric vector indicating the observation numbers to be included, or a character vector of the observation names that should be included. All observations are included by default.

na.rm : a logical flag: if `TRUE`, missing values will be removed before fitting the model. The default is `FALSE`.

contrasts : a list giving contrasts for some or all of the factors appearing in the model formula. An element in the list should have the same name as the factor variable it encodes, and it should be either a contrast matrix (any full-rank matrix with as many rows as there are levels in the factor), or a function that computes such a matrix given the number of levels.

start : a character string which can be passed to `timeDate` function to specify the starting date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

end : a character string which can be passed to `timeDate` function to specify the ending date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

`demean` : an optional argument only used if an intercept term is not included in the model. It can be a character string or a numeric vector which specifies how to demean the multiple time series. If `demean` is a character string, there are three valid choices: `"none"`, which does not demean the time series; `"all"`, which uses the mean of all observations to demean the time series; or `"sample"`, which uses the mean of the right hand side observations to demean the time series. If `demean` is a numeric vector, it must have the same length as the dimension of the multiple time series, and will be used to demean the time series. The default is `"none"`.

`...` : any optional arguments that can be passed to the `timeDate` function to parse the `start` or `end` specification.

VALUE

an object of class `"VAR"` representing the fit. See `VAR.object` for details.

REFERENCES

Lutkepohl, H. (1990). *Introduction to Multiple Time Series Analysis*, Springer-Verlag.

Hamilton, J. D. (1994). *Time Series Analysis*, Princeton University Press.

SEE ALSO

`BVAR, VAR ,VAR.object` .

EXAMPLE

```
# use a time series data frame
pol.mod = VAR(cbind(CP,GDP,U)~ar(6), data=policy.dat)

# use a matrix directly without the data argument
pol.mat = as.matrix(seriesData(policy.dat))
pol.mod = VAR(pol.mat~ar(6))
```

VAR.object Vector Autoregression Model Objects

DESCRIPTION
>
> These are objects of class `"VAR"` which represent the fit of a vector autoregression model.

GENERATION
>
> This class of objects is returned from the `VAR` function.

METHODS
>
> The `"VAR"` class of objects currently has methods for the following generic functions:
>
> `coef, residuals, fitted, vcov, formula, plot, predict, cpredict, print, summary`.

STRUCTURE
>
> The following components must be present in a legitimate `"VAR"` object:

`call` : an image of the call that produced the object, but with the arguments all named and with the actual formula included as the `formula` argument.

`coef` : a matrix giving the coefficients of the VAR model.

`residuals` : two-dimensional object representing the residuals from the fit.

`fitted` : two-dimensional object representing the fitted values from the fit.

`df.resid` : the number of degrees of freedom for residuals.

`Sigma` : the un-scaled error covariance matrix. It needs to be multiplied by the degree of freedom of residuals to produce the estimate of error covariance matrix.

`ar.order` : the order of the autoregressive model.

`R` : the triangular factor of the decomposition, which is determined by the orthogonal decomposition of the model matrix.

`rank` : the computed rank (number of linearly independent columns in the model matrix).

`Y0` : the first p rows of the original data, where p is the autoregressive order.

`contrasts` : a list containing sufficient information to construct the contrasts used to fit any factors occurring in the model. The list contains entries that are either matrices or character vectors. When a factor is coded by contrasts, the corresponding contrast matrix is stored in this list.

Factors that appear only as dummy variables and variables in the model that are matrices correspond to character vectors in the list. The character vector has the level names for a factor or the column labels for a matrix.

terms : an object of mode `expression` and class `term` summarizing the formula. Used by various methods, but typically not of direct relevance to users.

SEE ALSO
VAR.

VaR.plot Function to Plot the Adverse Movement for a Given Percentage

DESCRIPTION

Plots the predicted conditional density of the time series from a fitted object of class `"garch"` .

`VaR.plot(x, last=200, q.point=0.01)`

REQUIRED ARGUMENTS

x: an object of class `"garch"`, usually the fitted object from the function `garch()`.

OPTIONAL ARGUMENTS

last: the length of the last portion of the time series used to estimate the Gaussian distribution by the classical method.

q.point: the input percentage probability.

the classical Gaussian density estimated from the last portion of the time series and the predicted conditional density based on the GARCH model are plotted for comparison.

SEE ALSO

`plot.garch`.

EXAMPLE

```
hp.fit = garch(hp.s~1, ~garch(1,1))
VaR.plot(hp.fit)
```

VAR Fit a Vector Autoregression Model

DESCRIPTION

Returns an object of class `"VAR"` that represents the fit of a VAR model.

```
VAR(formula, data, subset, na.rm=F, contrasts=NULL, start=NULL,
    end=NULL, demean="none", max.ar=min(12, max.lag),
    criterion="BIC", ...)
```

REQUIRED ARGUMENTS

`formula` : a `formula` object, with the response on the left of a $\sim$ operator and the terms, separated by `+` operators, on the right. The response must be a `"matrix"` object if the `data` argument is not specified.

OPTIONAL ARGUMENTS

`data` : a data frame or `"timeSeries"` data frame in which to interpret the variables named in the `formula` and `subset` arguments. If `data` is missing, the variables in the model formula should be in the search path, and cannot be `"timeSeries"` objects.

`subset` : an expression specifying which subset of observations should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), a numeric vector indicating the observation numbers to be included, or a character vector of the observation names that should be included. All observations are included by default.

`na.rm` : a logical flag: if `TRUE`, missing values will be removed before fitting the model. The default is `FALSE`.

`contrasts` : a list giving contrasts for some or all of the factors appearing in the model formula. An element in the list should have the same name as the factor variable it encodes, and it should be either a contrast matrix (any full-rank matrix with as many rows as there are levels in the factor), or a function that computes such a matrix given the number of levels.

`start` : a character string which can be passed to `timeDate` function to specify the starting date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

`end` : a character string which can be passed to `timeDate` function to specify the ending date for the estimation. This can only be used if the `data` argument is a `"timeSeries"` data frame. The default is `NULL`.

`demean` : an optional argument only used if an intercept term is not included in the model. It can be a character string or a numeric vector which specifies how to demean the multiple time series. If `demean` is a character string, there are three valid choices: `"none"`, which does not demean the time series; `"all"`, which uses the mean of all observations to demean the time series; or `"sample"`, which uses the mean of the right hand side observations to demean the time series. If `demean` is a numeric vector, it must have the same length as the dimension of the multiple time series, and will be used to demean the time series. The default is `"none"`.

`max.ar` : an integer which specifies the maximal autoregressive order to consider for automatic lag selection. The default is the smaller of 12 and `max.lag`, where `max.lag` is defined as `(n-1)/(nvar+1)`, `n` being the number of time periods in `data`, and `nvar` the number of columns in `data`.

`criterion` : a character string which specifies the criterion for automatic lag selection. Valid choices are `"logL"` for log-likelihood value, `"aic"` for Akaike Information Criterion, `"bic"` for Bayesian Information Criterion, and `"hq"` for Hannan-Quinn information criterion. The default is `"bic"`.

`...` : any optional arguments that can be passed to the `timeDate` function to parse the `start` or `end` specification.

VALUE

an object of class `"VAR"` representing the fit. See `VAR.object` for details.

DETAILS

If `formula` is missing, then `data` must be supplied. In this case, an automatic lag selection procedure will be used to find the "best" model for all the variables in `data` according to a user-chosen information criterion. Otherwise, `VAR.formula` will be called to fit the model.

REFERENCES

Lutkepohl, H. (1990). *Introduction to Multiple Time Series Analysis*, Springer-Verlag.

Hamilton, J. D. (1994). *Time Series Analysis*, Princeton University Press.

SEE ALSO

`BVAR`, `VAR.formula` ,`VAR.object` .

EXAMPLE

```
# use a time series data frame
pol.mod = VAR(cbind(CP,GDP,U)~ar(6), data=policy.dat)
```

```
# use automatic model selection
VAR(policy.dat)
```

vcov.mfactor   Use vcov() on an mfactor Object

DESCRIPTION

> This is a method for the function vcov() for objects inheriting from
> class "mfactor". See vcov for the general behavior of this function
> and for the interpretation of object.

> vcov.mfactor(object)

REQUIRED ARGUMENTS

object : an object inheriting from class "mfactor".

VALUE

> the variance-covariance matrix of the original multivariate data ac-
> cording to the fitted multi-factor model.

SEE ALSO

> mfactor.object, vcov .

vcov.mgarch Covariance Matrix of GARCH/MGARCH/FGARCH
Parameters

DESCRIPTION

Returns a covariance matrix of the parameter estimates in GARCH/MGARCH/FGARCH models, as estimated by the`garch` , `mgarch` or `fgarch` function.

```
cov.mgarch(x, method="OP", partial=T)
cov.fgarch(x, method="OP", partial=T)
```

REQUIRED ARGUMENTS

x: an object of class `"garch"`, `"mgarch"`, or `"fgarch"`, as returned by `garch`, `mgarch` or `fgarch` function.

OPTIONAL ARGUMENTS

method: parameter that determines the type of covariance matrix returned. Valid choices are `"OP"` for outer product v ersion of the covariance matrix, `"HESSIAN"` for the covariance matrix calculated using the inverse of numerical Hessian matrix, and `"QMLE"` or `"ROBUST"` for the covariance matrix calculated using the asymptotic result for quasi-maximum likelihood estimation, sometimes referred to as the robust covariance matrix.

partial: a logical flag: if `FALSE`, standard deviations of other parameters, such as the degree of freedom for conditional t-distribution, if estimated, are also returned. The default is `TRUE`.

VALUE

the selected covariance matrix if `partial=T`, or a list with the following elements if `partial=F`: cov being the covariance matrix of GARCH/MGARCH/FGARCH parameters, `par.se` being the standard deviation of the distributional parameter if there is one and it is estimated, `cccor.se` being the standard deviations of the estimated constant conditional correlation parameters for CCC model.

REFERENCES

Bollerslev, T. P., and Wooldridge, J. M. (1992). Quasi-Maximum Likelihood Estimation of Dynamic Models with Time-Varying Covariances, *Econometric Reviews*, 11:143-172.

EXAMPLE

```
hp.cov = vcov(garch(hp.s~1, ~garch(1,1)))
```

vcov.OLS Use vcov() on an OLS Object

DESCRIPTION

This is a method for the function vcov() for objects inheriting from class "OLS". See vcov for the general behavior of this function and for the interpretation of object.

```
vcov.OLS(object, correction=NULL, bandwidth=NULL,
         window="bartlett", unbiased=T)
```

REQUIRED ARGUMENTS

object : an object inheriting from class "OLS".

OPTIONAL ARGUMENTS

correction : a character string specifying the type of correction. If NULL, no correction is performed and the classical covariance matrix is returned. If correction is "white" or "hc", then White's heteroskedasticity-consistent covariance matrix is returned. If correction is "nw" or "hac", then Newey-West's heteroskedasticity and autocorrelation consistent covariance matrix is returned. The default is NULL.

bandwidth : an integer that specifies the bandwidth to be used. This is only used when correction is "nw" or "hac". The default is set to $4*(n/100)^{0.25}$, where n is the sample size.

window : a character string that specifies the type of window to be used. Currently the only valid choices are "Bartlett" for Bartlett's triangular window, or "rectangular" for a rectangular window. This is only used when correction is "nw" or "hac". The default is "Bartlett".

unbiased : a logical flag; if TRUE, unbiased estimate of error covariance will be used; if FALSE, the maximum likelihood estimate of error covariance will be used. The default is TRUE.

VALUE

the variance-covariance matrix of estimated OLS coefficients.

REFERENCES

Newey, W. K., and West, K. D. (1987). A Simple, Positive Semi-definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, 55(3):703-08.

White, H. (1980), A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48:817-838.

SEE ALSO

autocorTest, heteroTest.OLS ,vcov .

vcov.rollOLS Use vcov() on a rollOLS Object

DESCRIPTION

This is a method for the function vcov() for objects inheriting from class "rollOLS". See vcov for the general behavior of this function and for the interpretation of object.

    vcov.rollOLS(object)

REQUIRED ARGUMENTS

object : an object inheriting from class "rollOLS".

VALUE

a matrix giving the variance of coefficient estimates with each row from each rolling sample.

SEE ALSO

rollOLS.object, vcov .

vcov Variance/Covariance Matrix of Model Parameter Estimates

DESCRIPTION

Returns the variance-covariance matrix of the parameter estimates given a fitted model object.

This function is an S Version 3 generic (see `Methods`); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: `OLS`, `VAR`, `mfactor`, `rollOLS`.

```
vcov(object, ...)
```

REQUIRED ARGUMENTS

`object` : any object representing a fitted model. It usually contains a component called `"cov"` or `"vcov"`.

VALUE

a matrix representing the covariance matrix.

vcov.VAR   Use vcov() on a VAR Object

DESCRIPTION

This is a method for the function vcov() for objects inheriting from class "VAR". See vcov for the general behavior of this function and for the interpretation of object.

    vcov.VAR(object, unbiased=T)

REQUIRED ARGUMENTS

object : an object inheriting from class "VAR".

OPTIONAL ARGUMENTS

unbiased : a logical flag; if TRUE, unbiased estimate of error covariance will be used; if FALSE, the maximum likelihood estimate of error covariance will be used. The default is TRUE.

VALUE

the variance-covariance matrix of estimated VAR coefficients.

SEE ALSO

vcov.

VECM2VAR VAR Representation of a VECM Object

DESCRIPTION

Returns the VAR representation of a VECM, given a fitted `"VECM"` object. This is called by `predict.VECM` to compute the forecasts from a VECM.

```
VECM2VAR(object)
```

REQUIRED ARGUMENTS

`object` : an object of class `"VECM"`, usually returned by a call to `VECM` function.

VALUE

a list containing the following components:

`var.coef` : a matrix of dimension `(p*k) x k` giving the autoregressive coefficients of the VAR representation, where `p` is the autoregressive order of the VAR representation, and `k` is the number of dependent variables.

`trend.coef` : a matrix of dimension `c x k` giving the coefficients of the trend term of the VAR representation if present, where `c` is 1 if there is only a constant or 2 if there is a linear trend.

`exo.coef` : a matrix giving the coefficients of the exogenous variables if present.

REFERENCES

Johansen, S. (1995). *Likelihood-based Inference in Cointegrated Vector Autoregressive Models*. Oxford University Press.

SEE ALSO

`VAR.ar2ma,FARIMA.d2ar` .

EXAMPLE

```
mk.coint = coint(mk.zero2[,1:5], lags=1, trend="rc")
mk.vecm = VECM(mk.coint, coint.rank=3)
VECM2VAR(mk.vecm)
```

VECM.object Vector Error Correction Model Objects

DESCRIPTION

> These are objects of class `"VECM"` which inherit from the class `"VAR"`
> and represent the fit of a vector error correction model.

GENERATION

> This class of objects is returned from the `VECM` function.

METHODS

> The `"VECM"` class of objects currently has methods for the following
> generic functions:
>
> `coef, residuals, fitted, vcov, plot, predict, cpredict, print,`
> `summary.`

STRUCTURE

> The following components must be present in a legitimate `"VECM"`
> object, in addition to components of a legitimate `"VAR"` object:

`beta.c` : a matrix of dimension `(k+c) x r` giving the normalized cointegrating
vectors, where `k` is the number of dependent variables, `c` is 1 if `trend`
is `"rc"` or `"rt"` and 0 otherwise, and `r` is the cointegrating rank.

`se.beta` : a matrix of dimension `(k+c) x r` giving the standard errors of the
MLE estimate `beta.c`.

`trend` : a character string giving the trend specification in the original `"coint`
object. See the help files for `coint` and `coint.object` for details.

`coint.res` : a matrix or `"timeSeries"` object with a matrix in the data slot,
representing the cointegrating residuals or disequilibrium errors.

`ar.order` : an integer giving the autoregressive order of the VECM. Note that
this is one less than the autoregressive order of the corresponding VAR
representation.

SEE ALSO

> `VAR.object`,`coint` ,`coint.object` .

VECM Fit a Vector Error Correction Model

DESCRIPTION

Estimates a Vector Error Correction Model (VECM) based on a fitted cointegration object.

```
VECM(test, coint.rank=1, unbiased=T)
```

REQUIRED ARGUMENTS

`test` : an object of class `"coint"`, usually returned by a call of `coint`.

OPTIONAL ARGUMENTS

`coint.rank` : an integer specifying the desired cointegrating rank. The default is set to 1.

`unbiased` : a logical flag: if `TRUE`, the unbiased residual covariance estimate is used. The default is `TRUE`.

VALUE

an object of class `"VECM"` representing the fit of VECM model with the desired cointegrating rank. See `VECM.object` for details.

REFERENCES

Johansen, S. (1995). *Likelihood-based Inference in Cointegrated Vector Autoregressive Models*. Oxford University Press.

SEE ALSO

`BVAR`,`VAR` ,`VECM.object` ,`coint` .

EXAMPLE

```
mk.coint = coint(mk.zero2[,1:5], lags=1, trend="rc")
VECM(mk.coint, coint.rank=3)
```

waldTest Linear Wald Test on Regression Coefficients

DESCRIPTION

Returns linear Wald test statistic on regression coefficients.

This function is an S Version 3 generic (see `Methods`); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: `OLS`.

```
waldTest(object, ...)
```

REQUIRED ARGUMENTS

`object` : an object which is usually returned by a model fitting function, such as an `"OLS"` object.

VALUE

an object of class `"waldTest"`. The following components must be present in a legitimate `"waldTest"` object:

`method` : a character string describing the test computed.

`n` : an integer giving the sample size used to compute the test.

`na` : an integer giving the number of missing values in the test sample.

`statistic` : the value of the test statistic.

`p.value` : the corresponding p-value of the test.

`distribution` : a character string giving the distribution of the test statistic under the null hypothesis.

`parameters` : the degree of freedom of `distribution` under the null hypothesis.

DETAILS

By default, `waldTest.OLS` uses the classical covariance matrix of coefficients to compute the Wald statistics. To use HC or HAC covariance matrix, simply pass the optional arguments taken by `vcov.OLS`.

SEE ALSO

`autocorTest`,`collinearTest` ,`heteroTest` ,`vcov.OLS` .

xygPlot Trellis x-y Plot with Grid Options

DESCRIPTION

> Generates a trellis object representing an x-y plot with grid options.

> `xygPlot(x, y, strip.text="", hgrid=F, vgrid=F, ...)`

REQUIRED ARGUMENTS

 x, y : a numeric vector or a `"timeSeries"` object with a numeric vector in
  the data slot.

OPTIONAL ARGUMENTS

 strip.text : a character string which will appear in the strip of the trellis
  plot. The default is "", which draws nothing in the strip.

 hgrid : a logical flag: if `TRUE`, horizontal grids will be drawn on the trellis
  plot. The default is `FALSE`.

 vgrid : a logical flag: if `TRUE`, vertical grids will be drawn on the trellis plot.
  The default is `FALSE`.

 ... : any other optional arguments that can be passed down to low level
  plot functions.

 VALUE

> an object of class `trellis`, which is automatically plotted by `print.trellis`.
> If this function is invoked directly without saving the returned object,
> a trellis device will be open to draw the plot if one is not available yet.

SEE ALSO

> rvfPlot, `trellis.args` , xyplot .

EXAMPLE

```
# Comparison of high and low prices.
xygPlot(msft.dat[,"High"], msft.dat[,"Low"], strip="High vs. Low",
        xlab="Low", ylab="High")
```

yhoo.df Intra-day Yahoo Stock Prices

SUMMARY
>
> This is a data frame representing daily transaction information of Yahoo stock, with the following six columns:

Date: a character string representing the trading date.

Open: the opening price of the corresponding day.

High: the highest price of the corresponding day.

Low: the lowest price of the corresponding day.

Close: the closing price of the corresponding day.

Volume: the trading volume of the corresponding day.