Package 'portes'

August 8, 2014

Type Package

Title Portmanteau Tests for Univariate and Multivariate Time Series Models

Version 2.1-3

Date 2014-08-07

Author Esam Mahdi, Ken Jinkun Xiao and A. Ian McLeod

Maintainer A. Ian McLeod <aim@stats.uwo.ca>

URL http://www.stats.uwo.ca/faculty/aim and http://site.iugaza.edu.ps/emahdi

Description This package contains a set of portmanteau diagnostic checks for univariate and multivariate time series.

Depends R (>= 2.14.0), parallel

Suggests fGarch, FitAR, FGN, TSA, vars, tseries, forecast, akima

LazyLoad yes

LazyData yes

Classification/ACM G.3, G.4, I.5.1

Classification/MSC 62M10, 91B84

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-08 07:36:01

R topics documented:

`portes-package` *Portmanteau Tests for Univariate and Multivariate Time Series Models*

Description

This package contains a set of portmanteau diagnostic checks for univariate and multivariate time series based on the asymptotic approximation distributions and the Monte-Carlo significance test. More details about the portmanteau test statistics are given in the online vignette of this package. It can be used for generating a simulated data from nonseasonal `ARIMA` or `VARIMA` models whith innovations from finite or infinite variances distributions. The simulated data may have deterministic terms, a constant drift and a time trend, with non-zero mean.

Details

Package:	portes
Type:	Package
Version:	2.1-2
Date:	2013-10-16
LazyLoad:	yes
LazyData:	yes
Depends:	`R (>= 2.14.0), parallel`

Suggests:	fGarch(V.2150.81), FitAR(V.1.92), FGN(V.1.5), TSA(V.0.99), vars(V.1.5-0), tseries(V.0.10-29), forecast(V.3.25), akima(V.0.5-7)
Classification/ACM:	G.3, G.4, I.5.1
Classification/MSC:	62M10, 91B84
License:	GPL (>= 2)

Main Function

The main function in this package, portest, is used with univariate and multivariate time series. It implements the Portmanteau test statistics, gvtest, BoxPierce, LjungBox, Hosking, and LiMcLeod based on two methods. The first method uses the Monte-Carlo techniques as described by Lin and McLeod (2006), Mahdi and McLeod (2012) and the second one uses the approximation asymptotic distribution. Originally, the generalized variance portmanteau test, gvtest, for univariate time series was derived by Pena and Rodriguez (2002) based on the gamma distribution. Lin and McLeod (2006) proposed the Monte-Carlo version of this test and Mahdi and McLeod (2012) extended both methods to the multivariate case. Simulation results suggest that the Monte-Carlo version of gvtest statistic is more accurate and powerful than its competitors proposed by Box and Pierce (1970), Ljung and Box (1978), and Pena and Rodriguez (2002, 2006) in the univariate time series and Hosking (1980) and Li and McLeod (1981) in the multivariate time series.

The powerful parallel computing framework facility is implemented in this function using the package parallel. This package handles running much larger chunks of computations in parallel and was first included in R 2.14.0 based on the work done for CRAN packages multicore and snow.

The default argument in portest function, nslaves=1, implements the Monte-Carlo test on PC with only one CPU. Set the argument nslaves equals to a positive integer number greater than 1, provided that the default argument MonteCarlo=TRUE is selected, the package parallel requires the Message Passing Interface, MPI, language to be properly installed and implemented by the parallel computing program MPICH2 in the I O environment system in which the portes package will run. Instructions to install and run MPICH2 is given in the link http://www.stats.uwo.ca/faculty/yu/Rmpi. When MonteCarlo=FALSE is selected, the test statistic selected from the argument test will be implemented based on the asymptotic approximation distribution. The default test statistic is the generalized variance test, gvtest.

Test for usual residuals and GARCH effects:
By setting the argument SquaredQ=TRUE in portest function, the portmanteau test using the asymptotic distribution approximation or the Monte-Carlo significance test (depending on the choice of the argument MonteCarlo whether FALSE or TRUE) for ARCH effects will be implemented on the squared residuals. Otherwise, the portmanteau test will be applied on the usual residuals (when the default argument SquaredQ=FALSE is selected). The goodness-of-fit garch model test can be implemented on fitted models with classes "garch" and "fGarch". These two classes are associated with output objects from the functions garch() and garchFit() available from the R packages tseries and fGarch respectively.

Monte-Carlo test for residuals with infinite variances:
The argument InfiniteVarianceQ=TRUE in portest function is used only with Monte-Carlo techniques. By selecting this argument, the Monte-Carlo diagnostic test on residuals with infinite variances effects is implemented.

Test for fractional Gaussian noise, FGN, effects:

After fitting FGN model using the function FitFGN() available from the FGN R package, the output object has a class "FitFGN". By substituting this object as a first entry in the portest function, the portmanteau test based on the Monte-Carlo or the asymptotic distribution method (depending on the choice of the argument MonteCarlo whether FALSE or TRUE) for FGN model will be implemented.

Goodness-of-fit test for any fitted model: The portmanteau test statistics implemented in the functions, gvtest, BoxPierce, LjungBox, Hosking, LiMcLeod, can be used for testing the adequacy of any fitted model using the Monte-Carlo significance test or the asymptotic aprroximation distribution test. More details with illustrative examples are given in the documentation of the main function portest.

Simulate data from nonseasonal ARIMA(p,d,q) or VARIMA(p,d,q)

The function varima.sim in this package is useful for simulating data from nonseasonal ARIMA or VARIMA of order (p, d, q) with or without deterministic terms (drift and trend). The innovations series can be given via the argument innov or may be generated from the selected distribution specified from the argument innov.dist. The argument d must entered as a nonnegative integer in the ARIMA case, whereas it must entered as a vector of k components $d_1, ..., d_k$ in the VARIMA case. d_i represents the difference lag need to be applied on series i. The components of the argument StableParameters are the stable parameters ALPHA, BETA, GAMMA, and DELTA needed to generate innovations from stable distribution.

Author(s)

Author: Esam Mahdi, Ken Jinkun Xiao and A. Ian McLeod.

Maintainer: A. Ian McLeod <aimcleod@uwo.ca>

References

Hosking, J. R. M. (1980). "The Multivariate Portmanteau Statistic". Journal of American Statistical Association, 75, 602-608.

Li, W. K. and McLeod, A. I. (1981). "Distribution of The Residual Autocorrelations in Multivariate ARMA Time Series Models". Journal of The Royal Statistical Society, Series B, 43, 231-239.

Lin, J.-W. and McLeod, A.I. (2006). "Improved Generalized Variance Portmanteau Test". Computational Statistics and Data Analysis 51, 1731-1738.

Lin, J.-W. and McLeod, A.I. (2008). "Portmanteau Tests for ARMA Models with Infinite Variance". Journal of Time Series Analysis, 29, 600-617.

Mahdi, E. and McLeod, A.I. (2012). "Improved Multivariate Portmanteau Test". Journal of Time Series Analysis, 33(2), 211-222.

McCulloch, J. H. (1986). "Simple Consistent Estimator of Stable Distribution Parameters". Commun. Statist.–Simula., 15(4), 1109-1136.

McLeod A.I, Li W.K (1983). "Distribution of the Residual Autocorrelation in Multivariate ARMA Time Series Models". Journal of Time Series Analysis, 4, 269-273.

McLeod, A.I., Yu, Hao, and Krougly, Z. L. (2007). "Algorithms for Linear Time Series Analysis". Journal of Statistical Software.

Pena, D. and Rodriguez, J. (2006). "The log of the determinant of the autocorrelation matrix for testing goodness of fit in time series". Journal of Statistical Planning and Inference, 136, 2706-2718.

Tierney, L., Rossini, A. J., Li, N., and Sevcikova, H. (2009). snow: Simple Network of Workstations. R package version 0.3-10. http://CRAN.R-project.org/package=snow.

Wuertz, D. and core team members R (2012). fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling. R package version 2150.81. http://CRAN.R-project.org/package=fGarch.

Yu, H. (2002). Rmpi: Parallel Statistical Computing in R. R News, 2(2), 10-14. http://CRAN.R-project.org/doc/Rnews.

Examples

```
## Not run:
##############################################################################
####                                                                      ####
####                   Monte-Carlo Portmanteau Tests                      ####
####                                                                      ####
##############################################################################
## Monte-Carlo test for randomness series                                   ##
##############################################################################
data("DEXCAUS")
returns <- log(DEXCAUS[-1]/DEXCAUS[-length(DEXCAUS)])
portest(returns)            ## MC using one CPU takes about 25.16 seconds
portest(returns, nslaves=4) ## MC using 4 CPUs takes about 9.51 seconds
portest(returns, MonteCarlo=FALSE)                ## asymptotic gvtest
portest(returns,test="LjungBox", MonteCarlo=FALSE) ## asymptotic LjungBox
##############################################################################
## Monte-Carlo goodness-of-fit arima test using 4 CPUs                     ##
##############################################################################
## arima() or Arima() function takes about 14.32 seconds
ans1 <- arima(WWWusage,order=c(3,1,0))
portest(ans1, nslaves = 4)
#
## arima0() function takes about 15.26 seconds
ans2 <- arima0(WWWusage,order=c(3,1,0))
portest(ans2, nslaves = 4)
#
## auto.arima() function from package forecast takes about 13.59 seconds
library("forecast")
ans3 <- auto.arima(WWWusage)
portest(ans3, nslaves = 4)
#
## ar() function takes about 9.39 seconds
ans4 <- ar(Nile,order.max=2)
portest(ans4, nslaves = 4)
#
## FitAR() function takes about 10.78 seconds
library("FitAR")
ans5 <- FitAR(Nile, p=2)
portest(ans5, nslaves = 4)
##############################################################################
```

```
## Monte-Carlo goodness-of-fit VAR test - Multivariate series     ##
#########################################################################
data("IbmSp500")
ibm <- log(IbmSp500[,2]+1)*100
sp500 <- log(IbmSp500[,3]+1)*100
IBMSP500 <- data.frame(cbind(ibm,sp500))
## ar.ols() function takes about 9.11 seconds
ans6 <-  ar.ols(IBMSP500, aic=FALSE, intercept=TRUE, order.max=5)
portest(ans6, NREP=100, test="gvtest", nslaves=4)
## VAR() function takes about 11.55 seconds
library("vars")
ans7 <- VAR(IBMSP500, p=5)
portest(ans7, NREP=100, test="gvtest", nslaves=4)
portest(ans7,test="Hosking", MonteCarlo=FALSE) ## asymptotic Hosking test
#########################################################################
## Monte-Carlo test for GARCH effects using 4 CPUs                     ##
#########################################################################
## Example 1
## Test for ARCH effects on returns series takes about 14.65 seconds
data("monthintel")
returns <- as.ts(monthintel)
lags <- c(5, 10, 20, 40)
portest(returns, lags = lags, nslaves = 4, SquaredQ = TRUE)
#
## Example 2
library("fGarch")
library("tseries")
data("GNPDEF")
z<-ts(GNPDEF[,2], start=1947, freq=4)
r <- 100*diff(log(z))
## use garch() function takes about 6.75 seconds
FitGarch1 <- garch(r, order = c(1,1))
portest(FitGarch1,NREP=100,nslaves = 4)
portest(FitGarch1,NREP=100,nslaves = 4,SquaredQ=TRUE)
#
## use garchFit() function takes about 13.56 seconds
GarchFit2 <- garchFit(formula = ~arma(4,0)+garch(1,1), data=r, trace=FALSE)
portest(GarchFit2, NREP=100, nslaves = 4, SquaredQ = FALSE)
portest(GarchFit2, NREP=100, nslaves = 4, SquaredQ = TRUE)
#########################################################################
## Monte-Carlo test on residuals with infinite variances               ##
#########################################################################
## It takes about 32.7 seconds on personal PC with 4 CPUs
data("CRSP")
CRSP.AR5<- arima(CRSP, c(5, 0, 0))
lags <- c(10, 20, 30)
portest(CRSP.AR5,lags=lags,nslaves=4,NREP=1000,InfiniteVarianceQ=TRUE)
#########################################################################
## Monte-Carlo test for Fractional Gaussian Noise, FGN.                ##
#########################################################################
## It takes about 55.06 seconds on personal PC with 4 CPUs
library("FGN")
data("NileMin")
```

```
NILE.FGN <- FitFGN(NileMin)
lags <- c(5, 10, 20)
portest(NILE.FGN, lags = lags, nslaves = 4)
portest(NILE.FGN, MonteCarlo=FALSE)  ## asymptotic distribution method
#################################################################
## Write two functions to fit a model and simulate results
## Apply Monte-Carlo test on fitted obj with class "list"
#################################################################
## Example 1
## Threshold Autoregressive (TAR) Model example from TSA package
## It takes about 64.27 seconds on personal PC with 4 CPUs
library("TSA")
FitModel <- function(data){
    fit <- TSA::tar(y=log(data),p1=4,p2=4,d=3,a=0.1,b=0.9,print=FALSE)
    res <- ts(fit$std.res)
    parSpec <- list(res=res,fit=fit)
  parSpec
}
SimModel <- function(parSpec){
    fit <- parSpec$fit
  exp(tar.sim(fit)$y)
}
data(prey.eq)
portest(FitModel(prey.eq),nslaves=4,func=list(SimModel,FitModel),pkg="TSA")
#
## Example 2
## It takes about 10.75 seconds on personal PC with 4 CPUs
FitModel <- function(data){
    fit <- ar(data,aic = FALSE, order.max=2)
    order <- 2
    res <- ts(fit$resid[-(1:order)])
    phi <- fit$ar
    theta <- NULL
    sigma <- fit$var.pred
    demean <- fit$x.mean
 list(res=res,phi=phi,theta=theta,sigma=sigma,demean=demean)
}
SimModel <- function(parSpec){
    res <- parSpec$res
    n <- length(res)
    innov <- sample(x=res,size=n,replace = TRUE)
    phi <- parSpec$phi
    theta <- parSpec$theta
    sigma <- parSpec$sigma
    demean <- parSpec$demean
   arima.sim(n = n, list(ar = phi, ma = theta), innov = innov,
             sd = sqrt(sigma), mean = demean)
}
Fit <- FitModel(Nile)
portest(Fit,nslaves=4,func=list(SimModel=SimModel,FitModel=FitModel),pkg="stats")
#############################################################################
####                                                                     ####
####                Simulation using varima.sim Function                 ####
```

```
####                                                                        ####
################################################################################
# Simulate MA(1) where innovation series is provided via argument innov
################################################################################
set.seed(1234)
n <- 200
phi <-  NULL
theta <-  0.6
d <- NA
sigma <- 1.9
Z <- varima.sim(phi, theta, d, sigma, n,innov=rnorm(n))
plot(Z)
################################################################################
# Simulate ARIMA(2,1,0) process with phi=c(1.3,-0.35), Gaussian innovations
# The series is truncated at lag 50
################################################################################
set.seed(1234)
Trunc.Series <- 40
n <- 1000
phi <- c(1.3, -0.35)
theta <- NULL
d <- 1
sigma <- 1
Z <- varima.sim(phi,theta,d,sigma,n,Trunc.Series=Trunc.Series)
coef(arima(Z,order=c(2,1,0)))
################################################################################
# Simulate MA(1) process with theta = 0.5, t5-distribution innovations
################################################################################
set.seed(1234)
n <- 200
phi <- NULL
theta <- 0.5
Z <- varima.sim(phi, theta, sigma=1, n=n, innov.dist="t", df=5)
plot(Z)
################################################################################
# Simulate univariate ARMA(2,1) process with length 500,
# phi = c(1.3, -0.35), theta = 0.1. Drift equation is 8 + 0.05*t
# Stable innovations with: ALPHA = 1.75, BETA = 0, GAMMA = 1, DELTA = 0
################################################################################
set.seed(1234)
n <- 500
phi <-  c(1.3, -0.35)
theta <-  0.1
constant <- 8
trend <- 0.05
demean <- 0
d <- 0
sigma <-  0.7
ALPHA <- 1.75
BETA <- 0
GAMMA <- 1
DELTA <- 0
Stable <- c(ALPHA,BETA,GAMMA,DELTA)
```

```
Z <- varima.sim(phi,theta,d,sigma,n,constant,trend,demean,
     innov.dist="stable",StableParameters=Stable)
plot(Z)
#########################################################################
# Simulate a bivariate white noise series from a multivariate t4-distribution
#########################################################################
set.seed(1234)
Z <- varima.sim(sigma=diag(2),n=200,innov.dist="t",df=4)
plot(Z)
#########################################################################
# Simulate a trivariate VARMA(1,1) process with length 300.
# phi = array(c(0.5,0.4,0.1,0.5,0,0.3,0,0,0.1), dim=c(k,k,1)), where k =3
# theta = array(c(0,0.25,0,0.5,0.1,0.4,0,0.25,0.6), dim=c(k,k,1)).
# innovations are generated from multivariate normal distribution
# The process have mean c(10, 0, 12),
# Drift equation a + b * t, where a = c(2,1,5), and b = c(0.01,0.06,0)
# The series is truncated at default value: Trunc.Series=ceiling(100/3)=34
#########################################################################
set.seed(1234)
k <- 3
n <- 300
Trunc.Series <-  50
phi <-  array(c(0.5,0.4,0.1,0.5,0,0.3,0,0,0.1),dim=c(k,k,1))
theta <-  array(c(0,0.25,0,0.5,0.1,0.4,0,0.25,0.6),dim=c(k,k,1))
sigma <- diag(k)
constant <- c(2,1,5)
trend <- c(0.01,0.06,0)
demean <- c(10,0,12)
Z <- varima.sim(phi, theta, d = 0,sigma, n, constant,trend,demean)
plot(Z)
#########################################################################
# Simulate a bivariate VARIMA(2,d,1) process with n=300, where d=(1,2).
# phi = array(c(0.5,0.4,0.1,0.5,0,0.3,0,0),dim=c(k,k,2)),
# theta = array(c(0,0.25,0,0), dim=c(k,k,1)).
# innovations are generated from multivariate normal
# The process have mean zero and no deterministic terms.
# The variance covariance is sigma = matrix(c(1,0.71,0.71,2),2,2).
# The series is truncated at default value: Trunc.Series=ceiling(100/3)=34
#########################################################################
set.seed(1234)
k <- 2
n <- 300
Trunc.Series <-  50
phi <-  array(c(0.5,0.4,0.1,0.5,0,0.3,0,0),dim=c(k,k,2))
theta <-  array(c(0,0.25,0,0),dim=c(k,k,1))
d <- c(1,2)
sigma <- matrix(c(1,0.71,0.71,2),k,k)
Z <- varima.sim(phi, theta, d, sigma, n)
plot(Z)
#########################################################################
# Simulate a bivariate VAR(1) process with length 600.
# Stable distribution: ALPHA=(1.3,1.6), BETA=(0,0.2), GAMMA=(1,1), DELTA=(0,0.2)
# The series is truncated at default value: Trunc.Series=min(100,200)=100
```

```
#########################################################################
set.seed(1234)
k <- 2
n <- 600
phi <- array(c(-0.2,-0.6,0.3,1.1),dim=c(k,k,1))
theta <- NULL
d <- NA
sigma <- matrix(c(1,0.71,0.71,2),k,k)
ALPHA <- c(1.3,1.6)
BETA <- c(0,0.2)
GAMMA <-c(1,1)
DELTA <-c(0,0.2)
Stable <- c(ALPHA,BETA,GAMMA,DELTA)
Z <- varima.sim(phi,theta,d,sigma,n,innov.dist="stable",StableParameters=Stable)
plot(Z)

## End(Not run)
```

BoxPierce *The Univariate-Multivariate Box and Pierce Portmanteau Test*

Description

The univariate or multivariate Box-Pierce (1970) portmanteau test.

Usage

```
BoxPierce(obj,lags=seq(5,30,5),order=0,SquaredQ=FALSE)
```

Arguments

obj	a univariate or multivariate series with class "numeric", "matrix", "ts", or ("mts" "ts"). It can be also an object of fitted time-series model with class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", or "fGARCH". obj may also an object with class "list" (see details and following example).
lags	vector of lag auto-cross correlation coefficients used for BoxPierce test.
order	needed for degrees of freedom of asymptotic chi-square distribution. If obj is an object with class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", "fGARCH", or "list" then no need to enter the value of order as it will be automatically determined. In general order equals to the number of estimated parameters in the fitted model.
SquaredQ	if TRUE then apply the test on the squared values. This checks for Autoregressive Conditional Heteroscedastic, ARCH, effects. When SquaredQ = FALSE, then apply the test on the usual residuals.

Details

However the portmanteau test statistic can be applied directly on the output objects from the built in R functions ar(), FitAR(), arima(), arim0(), Arima(), auto.arima(), VAR(), garch(), garchFit(), FitFGN(), etc, it works with output objects from any fitted model. In this case, users should write their own function to fit any model they want. The object obj represents the output of this function. This output must be a list with at least two outcomes: the fitted residual and the order of the fitted model (list(res = ..., order = ...)). See the following example with the function FitModel().

Value

The Box and Pierce univariate or multivariate test statistic with the associated p-values for different lags based on the asymptotic chi-square distribution with k^2(lags-order) degrees of freedom.

Author(s)

Esam Mahdi and A.I. McLeod.

References

Box, G.E.P. and Pierce, D.A. (1970). "Distribution of Residual Autocorrelation in Autoregressive-Integrated Moving Average Time Series Models". Journal of American Statistical Association, 65, 1509-1526.

See Also

acf, Box.test, LjungBox, Hosking, LiMcLeod, gvtest, portest, GetResiduals, tar

Examples

```
## Not run:
x <- rnorm(100)
BoxPierce(x)
#################################################################
## Measurements of the annual flow of the river Nile at Aswan
## from the years 1871 to 1970:
#################################################################
fit <- arima(Nile, c(1, 0, 1))
lags <- c(5, 10, 20)
## Apply the univariate test statistic on the fitted model
BoxPierce(fit, lags)               ## Correct
BoxPierce(fit, lags, order = 2)    ## Correct
## Apply the test statistic on the residuals and set order = 2
res <- resid(fit)
BoxPierce(res, lags)               ## Wrong
BoxPierce(res, lags, order = 2)    ## Correct
#################################################################
## Quarterly, west German investment, income, and consumption
## from first quarter of 1960 to fourth quarter of 1982:
#################################################################
data(WestGerman)
```

```
DiffData <- matrix(numeric(3 * 91), ncol = 3)
  for (i in 1:3)
    DiffData[, i] <- diff(log(WestGerman[, i]), lag = 1)
fit <- ar.ols(DiffData, intercept = TRUE, order.max = 2)
lags <- c(5,10)
## Apply the test statistic on the fitted model
BoxPierce(fit,lags)               ## Correct
## Apply the test statistic on the residuals where order = 2
res <- ts((fit$resid)[-(1:2), ])
BoxPierce(res,lags)               ## Wrong
BoxPierce(res,lags,order = 2)     ## Correct
################################################################
## Monthly log stock returns of Intel corporation data
## Test for ARCH Effects
################################################################
monthintel <- as.ts(monthintel)
BoxPierce(monthintel)                ## Usual test
BoxPierce(monthintel,SquaredQ=TRUE)  ## Test for ARCH effects
################################################################
## Write a function to fit a model
## Apply portmanteau test on fitted obj with class "list"
################################################################
## Example 1
library("FitAR")
FitModel <- function(data){
    fit <- FitAR(z=data,p=2)
    p <- length(fit$phiHat)
    order <- p
    res <- fit$res
 list(res=res,order=order)
}
Fit <- FitModel(Nile)
BoxPierce(Fit)
##
## Example 2
library("TSA")
FitModel <- function(data){
    fit <- TSA::tar(y=log(data),p1=4,p2=4,d=3,a=0.1,b=0.9,print=FALSE)
    res <- ts(fit$std.res)
    p1 <- fit$p1
    p2 <- fit$p2
    order <- max(p1, p2)
    parSpec <- list(res=res,order=order)
  parSpec
}
data(prey.eq)
Fit <- FitModel(prey.eq)
BoxPierce(Fit)

## End(Not run)
```

CRSP — *Monthly simple returns of the CRSP value-weighted index, 1926 to 1997*

Description

This data has been discussed by Tsay (2002, Ch.2, p.38 and 39) and Lin and McLeod (2008). There are 864 data values.

Usage

```
data(CRSP)
```

References

Lin, J.-W. and McLeod, A.I. (2008). "Portmanteau Tests for ARMA Models with Infinite Variance". Journal of Time Series Analysis, 29, 600-617.

Tsay, R. S. (2002). Analysis of Financial Time Series. Wiley, New York.

Examples

```
acf(CRSP)
fitstable(CRSP)
```

DEXCAUS — *Canada/US Foreign Exchanges Rates, Daily, Jan. 4, 1971 to Sept. 5, 1996.*

Description

There are 2513 data values.

Usage

```
data(DEXCAUS)
```

Details

Title: Canada / U.S. Foreign Exchange Rate Series ID: DEXCAUS Source: Board of Governors of the Federal Reserve System Release: H.10 Foreign Exchange Rates Seasonal Adjustment: Not Applicable Frequency: Daily Units: Canadian Dollars to One U.S. Dollar Date Range: 1971-01-04 to 2006-09-05 Last Updated: 2006-09-06 10:42 AM CT Notes: Noon buying rates in New York City for cable transfers payable in foreign currencies.

Source

http://research.stlouisfed.org/fred2/series/DEXCAUS

Examples

```
acf(DEXCAUS)
fitstable(DEXCAUS)
```

`fitstable` *Fit Parameters to Stable Distributions, McCulloch (1986)*

Description

The quantile method of McCullogh (1986).

Usage

```
fitstable(x)
```

Arguments

x	univariate or independent multivariate variables of dimension k.

Details

The quantile estimation method of McCulloch (1986) is used for each variable in x. It is highly reliable, fast and reasonably efficient especially bearing in mind that in most applications there is a lot of data.

Value

matrix of k rows and 4 columns. k represents the number of the variables in the vector x and the columns with named components alpha, beta, scale, and location respectively.

Author(s)

Esam Mahdi, A.I. McLeod, and Jen-Wen Lin

References

Lin, J.-W. and McLeod A.I.(2008). "Portmanteau Tests for ARMA Models with Infinite Variance." Journal of Time Series Analysis, 29, 600-617.

McCulloch, J. H. (1986). "Simple Consistent Estimator of Stable Distribution Parameters". Commun. Statist.–Simula., 15(4), 1109-1136.

See Also

There is also a function `stableFit()` in the `fBasics` package for fitting stable distributions for univariate data. See also `rStable`, `varima.sim`,

Examples

```
## Not run:
## Univariate
x <- rStable(800, 1.7, 0, 1, 0)
fitstable(x)
## Multivariate
ALPHA <- c(1.3,1.6)
BETA <- c(0,0.2)
GAMMA <-c(1,1)
DELTA <-c(0,0.2)
x <- rStable(500, ALPHA, BETA, GAMMA, DELTA)
fitstable(x)

## End(Not run)
```

`GetResiduals` *Extract Residuals from ARIMA, VAR, FGN, GARCH, or any Fitted Time Series Model*

Description

This utility function is useful to use in the portmanteau functions, `BoxPierce`, `gvtest`, `Hosking`, `LiMcLeod`, `LjungBox`, and `portest`. It takes a fitted time-series object with class `"ar"`, `"arima0"`, `"Arima"`, `"varest"`, `"FitAR"`, `"FitFGN"`, `"garch"`, `"fGARCH"`, or `"list"`. and returns the residuals and the order from the fitted object.

Usage

```
GetResiduals(obj)
```

Arguments

`obj`	a fitted time-series model with class `"ar"`, `"arima0"`, `"Arima"`, `"varest"`, `"FitAR"`, `"FitFGN"`, `"garch"`, `"fGARCH"`, or `"list"`.

Value

List of order of fitted time series model and residuals from this model.

Author(s)

Esam Mahdi and A.I. McLeod.

See Also

`BoxPierce`, `gvtest`, `Hosking`, `LiMcLeod`, `LjungBox`, `ar`, `arima0`, `arima`, `Arima`, `FitAR`, `VAR`, `FitFGN`, `garch`, `garchFit`, `tar`

Examples

```
fit <- arima(Nile, c(1, 0, 1))
GetResiduals(fit)
```

GNPDEF *GNP Deflator for U.S. Inflation Data from January 01, 1947 to April 01, 2010.*

Description

GNP deflator for U.S. inflation data from 1947-01-01 to 2010-04-01.

Usage

```
data(GNPDEF)
```

Format

A data frame with 254 observations on the following 2 variables.

`time` time

`GNPDEF` a numeric vector denotes the GNP deflator

Source

http://research.stlouisfed.org

References

Bollerslev, T. (1986). "Generalized autoregressive conditional heteroskedasticity". Journal of Econometrics, 31(3), 307-327.

Examples

```
plot(ts(GNPDEF[,2]))
```

gvtest *Generalized Variance Portmanteau Test*

Description

New generalized variance portmanteau test based on the determinant of the Hosking's autocorrelation block Toeplitz matrix with order $m + 1$ given in the function ToeplitzBlock, where m represents the order of the block matrix.

Usage

```
gvtest(obj,lags=seq(5,30,5),order=0,SquaredQ=FALSE,Kernel=FALSE)
```

Arguments

obj	a univariate or multivariate series with class "numeric", "matrix", "ts", or ("mts" "ts"). It can be also an object of fitted time-series model with class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", or "fGARCH". obj may also an object with class "list" (see details and following example).
lags	vector of lag auto-cross correlation coefficients used for gvtest test.
order	needed for degrees of freedom of asymptotic chi-square distribution. If obj is an object with class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", "fGARCH", or "list" then no need to enter the value of order as it will be automatically determined. In general order equals to the number of estimated parameters in the fitted model.
SquaredQ	if TRUE then apply the test on the squared values. This checks for Autoregressive Conditional Heteroscedastic, ARCH, effects. When SquaredQ = FALSE, then apply the test on the usual residuals.
Kernel	logic. If TRUE then Parzen Kernel would be applied on the autocorrelation function for statistics.

Details

However the portmanteau test statistic can be applied directly on the output objects from the built in R functions ar(), FitAR(), arima(), arim0(), Arima(), auto.arima(), VAR(), garch(), garchFit(), FitFGN(), etc, it works with output objects from any fitted model. In this case, users should write their own function to fit any model they want. The object obj represents the output of this function. This output must be a list with at least two outcomes: the fitted residual and the order of the fitted model (list(res = ..., order = ...)). See the following example with the function FitModel().

Value

The generalized variance portmanteau test statistic and its associated p-values for different lags based on asymptotic chi-square as given in Mahdi and McLeod (2012).

Author(s)

Esam Mahdi, Ken Jinkun Xiao and A.I. McLeod.

References

Mahdi, E. and McLeod, A.I. (2012). "Improved Multivariate Portmanteau Test". Journal of Time Series Analysis, 33(2), 211-222.

Pena, D. and Rodriguez, J. (2002). "A Powerful Portmanteau Test of Lack of Test for Time Series". Journal of American Statistical Association, 97, 601-610.

Pena, D. and Rodriguez, J. (2006). "The log of the determinant of the autocorrelation matrix for testing goodness of fit in time series". Journal of Statistical Planning and Inference, 136, 2706-2718.

See Also

acf, Box.test, BoxPierce, LjungBox, Hosking, LiMcLeod, portest, ToeplitzBlock, GetResiduals, tar

Examples

```
## Not run:
x <- rnorm(100)
gvtest(x)
#################################################################
## Measurements of the annual flow of the river Nile at Aswan
## from the years 1871 to 1970:
#################################################################
fit <- arima(Nile, c(1, 0, 1))
lags <- c(5, 10, 20, 30)
## Apply the univariate test statistic on the fitted model
gvtest(fit, lags)               ## Correct
gvtest(fit, lags, order = 2)    ## Correct
## Apply the test statistic on the residuals and set order = 2
res <- resid(fit)
gvtest(res, lags)               ## Wrong
gvtest(res, lags, order = 2)    ## Correct
#################################################################
## Quarterly, west German investment, income, and consumption
## from first quarter of 1960 to fourth quarter of 1982:
#################################################################
data(WestGerman)
DiffData <- matrix(numeric(3 * 91), ncol = 3)
  for (i in 1:3)
    DiffData[, i] <- diff(log(WestGerman[, i]), lag = 1)
fit <- ar.ols(DiffData, intercept = TRUE, order.max = 2)
lags <- seq(5,30,5)
## Apply the test statistic on the fitted model
gvtest(fit,lags)                ## Correct
## Apply the test statistic on the residuals where order = 2
res <- ts((fit$resid)[-(1:2), ])
gvtest(res,lags)                ## Wrong
gvtest(res,lags,order = 2)      ## Correct
```

```
#################################################################
## Monthly log stock returns of Intel corporation data
## Test for ARCH Effects
#################################################################
monthintel <- as.ts(monthintel)
gvtest(monthintel)               ## Usual test
gvtest(monthintel,SquaredQ=TRUE)  ## Test for ARCH effects
#################################################################
## Write a function to fit a model
## Apply portmanteau test on fitted obj with class "list"
#################################################################
## Example 1
library("FitAR")
FitModel <- function(data){
    fit <- FitAR(z=data,p=2)
    p <- length(fit$phiHat)
    order <- p
    res <- fit$res
 list(res=res,order=order)
}
Fit <- FitModel(Nile)
gvtest(Fit)
##
## Example 2
library("TSA")
FitModel <- function(data){
    fit <- TSA::tar(y=log(data),p1=4,p2=4,d=3,a=0.1,b=0.9,print=FALSE)
    res <- ts(fit$std.res)
    p1 <- fit$p1
    p2 <- fit$p2
    order <- max(p1, p2)
    parSpec <- list(res=res,order=order)
  parSpec
}
data(prey.eq)
Fit <- FitModel(prey.eq)
gvtest(Fit)

## End(Not run)
```

Hosking *The Modified Multivariate Portmanteau Test, Hosking (1980)*

Description

The modified multivariate portmanteau test suggested by Hosking (1980).

Usage

```
Hosking(obj,lags=seq(5,30,5),order=0,SquaredQ=FALSE)
```

Arguments

obj	a univariate or multivariate series with class "numeric", "matrix", "ts", or ("mts" "ts"). It can be also an object of fitted time-series model with class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", or "fGARCH". obj may also an object with class "list" (see details and following example).
lags	vector of lag auto-cross correlation coefficients used for Hosking test.
order	needed for degrees of freedom of asymptotic chi-square distribution. If obj is an object with class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", "fGARCH", or "list" then no need to enter the value of order as it will be automatically determined. In general order equals to the number of estimated parameters in the fitted model.
SquaredQ	if TRUE then apply the test on the squared values. This checks for Autoregressive Conditional Heteroscedastic, ARCH, effects. When SquaredQ = FALSE, then apply the test on the usual residuals.

Details

However the portmanteau test statistic can be applied directly on the output objects from the built in R functions ar(), FitAR(), arima(), arim0(), Arima(), auto.arima(), VAR(), garch(), garchFit(), FitFGN(), etc, it works with output objects from any fitted model. In this case, users should write their own function to fit any model they want. The object obj represents the output of this function. This output must be a list with at least two outcomes: the fitted residual and the order of the fitted model (list(res = ..., order = ...)). See the following example with the function FitModel().

Value

The multivariate test statistic suggested by Hosking (1980) and its associated p-values for different lags based on the asymptotic chi-square distribution with k^2(lags-order) degrees of freedom.

Author(s)

Esam Mahdi and A.I. McLeod.

References

Hosking, J. R. M. (1980). "The Multivariate Portmanteau Statistic". Journal of American Statistical Association, 75, 602-608.

See Also

acf, Box.test, BoxPierce, LjungBox, LiMcLeod, gvtest, portest, GetResiduals, tar

Examples

```
## Not run:
#############################################################
## Quarterly, west German investment, income, and consumption
## from first quarter of 1960 to fourth quarter of 1982:
```

```
#################################################################
data(WestGerman)
DiffData <- matrix(numeric(3 * 91), ncol = 3)
  for (i in 1:3)
    DiffData[, i] <- diff(log(WestGerman[, i]), lag = 1)
fit <- ar.ols(DiffData, intercept = TRUE, order.max = 2)
lags <- c(5,10)
## Apply the test statistic on the fitted model
Hosking(fit,lags,order = 2)        ## Correct
Hosking(fit,lags)                  ## Correct
## Apply the test statistic on the residuals
res <- ts((fit$resid)[-(1:2), ])
Hosking(res,lags,order = 2)        ## Correct
Hosking(res,lags)                  ## Wrong
#################################################################
## Write a function to fit a model
## Apply portmanteau test on fitted obj with class "list"
#################################################################
## Example 1
FitModel <- function(data){
    fit <- ar.ols(data, intercept = TRUE, order.max = 2)
    order <- 2
    res <- res <- ts((fit$resid)[-(1:2), ])
 list(res=res,order=order)
}
Fit <- FitModel(DiffData)
Hosking(Fit)
##
## Example 2
library("TSA")
FitModel <- function(data){
    fit <- TSA::tar(y=log(data),p1=4,p2=4,d=3,a=0.1,b=0.9,print=FALSE)
    res <- ts(fit$std.res)
    p1 <- fit$p1
    p2 <- fit$p2
    order <- max(p1, p2)
    parSpec <- list(res=res,order=order)
  parSpec
}
data(prey.eq)
Fit <- FitModel(prey.eq)
Hosking(Fit)

## End(Not run)
```

IbmSp500 *Monthly Returns of IBM and S&P 500 Index*

Description

The monthly returns of IBM stock and the S&P 500 index from January 1926 to December 2008. This data has been discussed by Tsay (2010, Chapter 8).

Usage

```
data(IbmSp500)
```

Format

A data frame with 996 observations on the following 3 variables.

`date` a numeric vector

`ibm` a numeric vector

`sp` a numeric vector

Source

http://faculty.chicagobooth.edu/ruey.tsay/teaching/fts3/

References

Tsay, R. S. (2010). "Analysis of Financial Time Series". Wiley, New York, 3rd edition.

Examples

```
data(IbmSp500)
plot(IbmSp500)
acf(IbmSp500)
fitstable(IbmSp500)
```

ImpulseVMA *The Impulse Response Function in the Infinite MA or VMA Representation*

Description

The impulse coefficients are computed.

Usage

```
ImpulseVMA(phi=NULL,theta=NULL,Trunc.Series=NA)
```

Arguments

phi	a numeric or an array of AR or an array of VAR parameters with order p.
theta	a numeric or an array of MA or an array of VMA parameters with order q.
Trunc.Series	truncation lag is used to truncate the infinite MA or VMA Process. IF it is NA, then by default Trunc.Series=$p+q$.

Value

The impulse response coefficients of order Trunc.Series+1 obtained by converting the ARMA(p,q) or VARMA(p,q) process to infinite MA or VMA process, respectively.

Author(s)

Esam Mahdi and A.I. McLeod.

References

Lutkepohl, H. (2005). "New introduction to multiple time series analysis". Springer-Verlag, New York.

Reinsel, G. C. (1997). "Elements of Multivariate Time Series Analysis". Springer-Verlag, 2nd edition.

See Also

ARMAtoMA, varima.sim, vma.sim, InvertQ, InvertibleQ

Examples

```
## Not run:
#########################################################################
### Impulse response coefficients from AR(1,1) to infinite MA process.
### The infinite process is truncated at lag 20
###
k <- 1
Trunc.Series <- 20
phi <- 0.7
theta <- array(-0.9,dim=c(k,k,1))
ImpulseVMA(phi,theta,Trunc.Series)
#########################################################################
### Impulse response coefficients from VAR(2) to infinite VMA process
### The infinite process is truncated at default lag value = p+q
###
k <- 2
phi <- array(c(0.5,0.4,0.1,0.5,0,0.3,0,0),dim=c(k,k,2))
theta <- NULL
ImpulseVMA(phi,theta)
#########################################################################
### Impulse response coefficients from VARMA(2,1) to infinite VMA process
### The infinite process is truncated at lag 50
###
```

```
k <- 2
phi <- array(c(0.5,0.4,0.1,0.5,0,0.25,0,0),dim=c(k,k,2))
theta <- array(c(0.6,0,0.2,0.3),dim=c(k,k,1))
ImpulseVMA(phi,theta,Trunc.Series=50)

## End(Not run)
```

`InvertQ` *Check Stationary and Invertibility of ARMA or VARMA Models*

Description

Utility function checks whether `ARMA` or `VARMA` model satisfies the stationary or/and the invertibility conditions.

Usage

```
InvertQ(coef)
```

Arguments

`coef`	a numeric, matrix, or array.

Details

It should be noted that, the `AR`(p) or `VAR`(p) model can always be expressed as a kp-dimensional `AR`(1) or `VAR`(1), and the `MA`(q) or `VMA`(q) model can always be expressed as a kq-dimensional `MA`(1) or `VMA`(1). For this reason, we can use this fact when we need to find the explicit solutions of `AR`(p) or `VAR`(p) models or `MA`(q) or `VMA`(q) models as the `AR`(1) or `VAR`(1) or the `MA`(1) or `VMA`(1) models can be characterized with simple intuitive formulas.

Value

A warning message only if the model is not stationary or/and not invertible.

Author(s)

Esam Mahdi and A.I. McLeod.

References

Lutkepohl, H. (2005). "New introduction to multiple time series analysis". Springer-Verlag, New York.

Reinsel, G. C. (1997). "Elements of Multivariate Time Series Analysis". Springer-Verlag, 2nd edition.

See Also

`varima.sim`, `vma.sim`, `ImpulseVMA`

Examples

```
############################################################
### Check Stationary
phi <- array(c(0.5,0.4,0.1,0.5,0,0.3,0,0),dim=c(2,2,2))
InvertQ(phi)
### Check Invertibility
theta <- array(c(0.5,0.4,0.1,0.5,0,0.3,0,0),dim=c(2,2,2))
InvertQ(theta)
```

LiMcLeod *The Modified Multivariate Portmanteau Test, Li-McLeod (1981)*

Description

The modified multivariate portmanteau test suggested by Li and McLeod (1981).

Usage

```
LiMcLeod(obj,lags=seq(5,30,5),order=0,SquaredQ=FALSE)
```

Arguments

obj	a univariate or multivariate series with class "numeric", "matrix", "ts", or ("mts" "ts"). It can be also an object of fitted time-series model with class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", or "fGARCH". obj may also an object with class "list" (see details and following example).
lags	vector of lag auto-cross correlation coefficients used for LiMcLeod test.
order	needed for degrees of freedom of asymptotic chi-square distribution. If obj is an object with class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", "fGARCH", or "list" then no need to enter the value of order as it will be automatically determined. In general order equals to the number of estimated parameters in the fitted model.
SquaredQ	if TRUE then apply the test on the squared values. This checks for Autoregressive Conditional Heteroscedastic, ARCH, effects. When SquaredQ = FALSE, then apply the test on the usual residuals.

Details

However the portmanteau test statistic can be applied directly on the output objects from the built in R functions ar(), FitAR(), arima(), arim0(), Arima(), auto.arima(), VAR(), garch(), garchFit(), FitFGN(), etc, it works with output objects from any fitted model. In this case, users should write their own function to fit any model they want. The object obj represents the output of this function. This output must be a list with at least two outcomes: the fitted residual and the order of the fitted model (list(res = ..., order = ...)). See the following example with the function FitModel().

Value

The multivariate test statistic suggested by Li and McLeod (1981) and its corresponding p-values for different lags based on the asymptotic chi-square distribution with `k^2(lags-order)` degrees of freedom.

Author(s)

Esam Mahdi and A.I. McLeod.

References

Li, W. K. and McLeod, A. I. (1981). "Distribution of The Residual Autocorrelations in Multivariate ARMA Time Series Models". Journal of The Royal Statistical Society, Series B, 43, 231-239.

See Also

`acf`, `Box.test`, `BoxPierce`, `LjungBox`, `Hosking`, `gvtest`, `portest`, `GetResiduals`, `tar`

Examples

```
## Not run:
#############################################################
## Quarterly, west German investment, income, and consumption
## from first quarter of 1960 to fourth quarter of 1982:
#############################################################
data(WestGerman)
DiffData <- matrix(numeric(3 * 91), ncol = 3)
  for (i in 1:3)
    DiffData[, i] <- diff(log(WestGerman[, i]), lag = 1)
fit <- ar.ols(DiffData, intercept = TRUE, order.max = 2)
lags <- c(5,10)
## Apply the test statistic on the fitted model
LiMcLeod(fit,lags,order = 2)        ## Correct
LiMcLeod(fit,lags)                  ## Correct
## Apply the test statistic on the residuals
res <- ts((fit$resid)[-(1:2), ])
LiMcLeod(res,lags,order = 2)        ## Correct
LiMcLeod(res,lags)                  ## Wrong
#############################################################
## Write a function to fit a model
## Apply portmanteau test on fitted obj with class "list"
#############################################################
## Example 1
FitModel <- function(data){
    fit <- ar.ols(data, intercept = TRUE, order.max = 2)
    order <- 2
    res <- res <- ts((fit$resid)[-(1:2), ])
 list(res=res,order=order)
}
Fit <- FitModel(DiffData)
LiMcLeod(Fit)
```

```
##
## Example 2
library("TSA")
FitModel <- function(data){
    fit <- TSA::tar(y=log(data),p1=4,p2=4,d=3,a=0.1,b=0.9,print=FALSE)
    res <- ts(fit$std.res)
    p1 <- fit$p1
    p2 <- fit$p2
    order <- max(p1, p2)
    parSpec <- list(res=res,order=order)
  parSpec
}
data(prey.eq)
Fit <- FitModel(prey.eq)
LiMcLeod(Fit)

## End(Not run)
```

LjungBox *Ljung and Box Portmanteau Test*

Description

The Ljung-Box (1978) modified portmanteau test. In the multivariate time series, this test statistic is asymptotically equal to Hosking.

Usage

```
LjungBox(obj,lags=seq(5,30,5),order=0,SquaredQ=FALSE)
```

Arguments

obj	a univariate or multivariate series with class "numeric", "matrix", "ts", or ("mts" "ts"). It can be also an object of fitted time-series model with class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", or "fGARCH". obj may also an object with class "list" (see details and following example).
lags	vector of lag auto-cross correlation coefficients used for LjungBox test.
order	needed for degrees of freedom of asymptotic chi-square distribution. If obj is an object with class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", "fGARCH", or "list" then no need to enter the value of order as it will be automatically determined. In general order equals to the number of estimated parameters in the fitted model.
SquaredQ	if TRUE then apply the test on the squared values. This checks for Autoregressive Conditional Heteroscedastic, ARCH, effects. When SquaredQ = FALSE, then apply the test on the usual residuals.

Details

However the portmanteau test statistic can be applied directly on the output objects from the built in R functions ar(), FitAR(), arima(), arim0(), Arima(), auto.arima(), VAR(), garch(), garchFit(), FitFGN(), etc, it works with output objects from any fitted model. In this case, users should write their own function to fit any model they want. The object obj represents the output of this function. This output must be a list with at least two outcomes: the fitted residual and the order of the fitted model (list(res = ..., order = ...)). See the following example with the function FitModel().

Value

The Ljung and Box test statistic with the associated p-values for different lags based on the asymptotic chi-square distribution with k^2(lags-order) degrees of freedom.

Author(s)

Esam Mahdi and A.I. McLeod.

References

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

acf, Box.test, BoxPierce, Hosking, LiMcLeod, gvtest, portest, GetResiduals, tar

Examples

```
x <- rnorm(100)
LjungBox(x)
#################################################################
## Measurements of the annual flow of the river Nile at Aswan
## from the years 1871 to 1970:
#################################################################
fit <- arima(Nile, c(1, 0, 1))
lags <- c(5, 10, 20)
## Apply the univariate test statistic on the fitted model
LjungBox(fit, lags)               ## Correct
LjungBox(fit, lags, order = 2)    ## Correct
## Apply the test statistic on the residuals and set order = 2
res <- resid(fit)
LjungBox(res, lags)               ## Wrong
LjungBox(res, lags, order = 2)    ## Correct
#to save time the other examples are not run
## Not run:
#################################################################
## Quarterly, west German investment, income, and consumption
## from first quarter of 1960 to fourth quarter of 1982:
#################################################################
data(WestGerman)
```

```
DiffData <- matrix(numeric(3 * 91), ncol = 3)
  for (i in 1:3)
    DiffData[, i] <- diff(log(WestGerman[, i]), lag = 1)
fit <- ar.ols(DiffData, intercept = TRUE, order.max = 2)
lags <- c(5,10)
## Apply the test statistic on the fitted model
LjungBox(fit,lags)               ## Correct
## Apply the test statistic on the residuals where order = 2
res <- ts((fit$resid)[-(1:2), ])
LjungBox(res,lags)               ## Wrong
LjungBox(res,lags,order = 2)     ## Correct
################################################################
## Monthly log stock returns of Intel corporation data
## Test for ARCH Effects
################################################################
monthintel <- as.ts(monthintel)
LjungBox(monthintel)               ## Usual test
LjungBox(monthintel,SquaredQ=TRUE)  ## Test for ARCH effects
################################################################
## Write a function to fit a model
## Apply portmanteau test on fitted obj with class "list"
################################################################
## Example 1
library("FitAR")
FitModel <- function(data){
    fit <- FitAR(z=data,p=2)
    p <- length(fit$phiHat)
    order <- p
    res <- fit$res
 list(res=res,order=order)
}
Fit <- FitModel(Nile)
LjungBox(Fit)
##
## Example 2
library("TSA")
FitModel <- function(data){
    fit <- TSA::tar(y=log(data),p1=4,p2=4,d=3,a=0.1,b=0.9,print=FALSE)
    res <- ts(fit$std.res)
    p1 <- fit$p1
    p2 <- fit$p2
    order <- max(p1, p2)
    parSpec <- list(res=res,order=order)
  parSpec
}
data(prey.eq)
Fit <- FitModel(prey.eq)
LjungBox(Fit)

## End(Not run)
```

monthintel — *The Monthly Log Stock Returns of Intel Corporation from January 1973 to December 2003*

Description

The monthly log stock returns of Intel Corporation from January 1973 to December 2003. This data has been discussed by Tsay (2005, p.99-102). There are 372 data values.

Usage

```
data(monthintel)
```

Source

http://faculty.chicagobooth.edu/ruey.tsay/teaching/fts2/

References

Tsay, R. S. (2005). "Analysis of Financial Time Series". Wiley, New York, 2nd edition.

Examples

```
acf(monthintel)
fitstable(monthintel)
```

portest — *Portmanteau Test Statistics*

Description

Univariate or multivariate portmanteau test statistics based on the Monte-Carlo techniques or asymptotic distributions.

Usage

```
portest(obj,lags=seq(5,30,5),order=0,test=c("PenaRodriguez","BoxPierce",
  "LjungBox","Hosking","LiMcLeod"),MonteCarlo=TRUE,Kernel=FALSE,nworkers=1,
   NREP=1000,InfiniteVarianceQ=FALSE,SquaredQ=FALSE,
   func=list(SimModel=NULL,FitModel=NULL),pkg=NULL,SetSeed=TRUE)
```

Arguments

obj	if obj is an object of class "ar", "arima0", "Arima", "varest", "FitAR", "FitFGN", "garch", "fGARCH", or "list" then a portmanteau goodness-of-fit test is done on the fitted model. Otherwise, for obj with class "ts", "numeric", "matrix", or ("mts" "ts"), a test of randomness is done.
lags	vector of lag values is used for portmanteau test.
order	as described in BoxPierce, gvtest, Hosking, LiMcLeod, and LjungBox and needed only when MonteCarlo = FALSE is selected.
test	portmanteau test statistic type
MonteCarlo	if TRUE then apply the Monte-Carlo version of portmanteau statistic. Otherwise, apply the asymptotic distribution.
Kernel	Applicable when MonteCarlo=TRUE.Default=FALSE. If TRUE then Parzen Weight would be applied on the autocorrelation function for the wild bootstrap Monte Carlo Test.
nworkers	number of slaves needed to use in parallel calculations. Default is one single CPU.
NREP	number of replications needed for Monte-Carlo test.
InfiniteVarianceQ	FALSE, assumes innovations with finite variance. Otherwise, innovations follow stable distribution with infinite variance.
SquaredQ	as described in BoxPierce, PenaRodriguez, Hosking, LiMcLeod, and LjungBox.
func	additional argument defined as a list with two specified functions, SimModel and FitModel. This argument is needed when the class of obj is "list" (see details and following example).
pkg	the name of the library to be loaded if the Monte-Carlo significance test is used with an object obj with class "list".
SetSeed	if TRUE then set.seed is initialized.

Details

The portmanteau test statistics, gvtest, BoxPierce, LjungBox, Hosking, and LiMcLeod are implemented based on the Monte-Carlo techniques and the approximation asymptotic distributions as described in Mahdi and McLeod (2012). The null hypothesis assuming that the fitted model is an adequate model and the residuals behave like white noise series. The parallel computing program MPICH2 must be properly installed in the I O environment system in which the portes package will run if one decide to choose the argument MonteCarlo=TRUE provided that nworkers equals to a positive integer more than 1. This function can be used for testing the adequacy in the non-seasonal fitted ARIMA, VAR, and Fractional Gaussian Noise, FGN, models. Also, it can be used to check for randomness as well as for GARCH effects. Any other fitted model, for example, threashold autoregression model, may also be tested for adequacy. In this case, two functions, SimModel() and FitModel(), must be provided via the argument func. The object obj is the output of the fitted model coded in the function FitModel and it is a "list" with at least res, the residuals from the fitted model in FitModel(). The output from the function SimModel() is a simulated univariate or multivariate series from the fitted model coded in the function FitModel(). The argument pkg represents the name of the R package where the fitted model build in (see the last two examples).

Value

The portmanteau test statistic with the associated p-values for different lag values.

Author(s)

Esam Mahdi, Ken Jinkun Xiao and A.I. McLeod.

References

Box, G.E.P. and Pierce, D.A. (1970). "Distribution of Residual Autocorrelation in Autoregressive-Integrated Moving Average Time Series Models". Journal of American Statistical Association, 65, 1509-1526.

Hosking, J. R. M. (1980). "The Multivariate Portmanteau Statistic". Journal of American Statistical Association, 75, 602-608.

Li, W. K. and McLeod, A. I. (1981). "Distribution of The Residual Autocorrelations in Multivariate ARMA Time Series Models". Journal of The Royal Statistical Society, Series B, 43, 231-239.

Lin, J.-W. and McLeod, A.I. (2006). "Improved Generalized Variance Portmanteau Test". Computational Statistics and Data Analysis 51, 1731-1738.

Lin, J.-W. and McLeod, A.I. (2008). "Portmanteau Tests for ARMA Models with Infinite Variance". Journal of Time Series Analysis, 29, 600-617.

Ljung, G.M. and Box, G.E.P (1978). "On a Measure of Lack of Fit in Time Series Models". Biometrika, 65, 297-303.

Mahdi, E. and McLeod, A.I. (2012). "Improved Multivariate Portmanteau Test". Journal of Time Series Analysis, 33(2), 211-222.

McLeod A.I, Li W.K (1983). "Distribution of the Residual Autocorrelation in Multivariate ARMA Time Series Models". Journal of Time Series Analysis, 4, 269-273.

Pena, D. and Rodriguez, J. (2006). "The log of the determinant of the autocorrelation matrix for testing goodness of fit in time series". Journal of Statistical Planning and Inference, 136, 2706-2718.

Tierney, L., Rossini, A. J., Li, N., and Sevcikova, H. (2009). snow: Simple Network of Workstations. R package version 0.3-10. http://CRAN.R-project.org/package=snow.

Wuertz, D. and core team members R (2012). fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling. R package version 2150.81. http://CRAN.R-project.org/package=fGarch.

Yu, H. (2002). Rmpi: Parallel Statistical Computing in R. R News, 2(2), 10-14. http://CRAN.R-project.org/doc/Rnews.

See Also

varima.sim, ar, arima0, arima, Arima, auto.arima, FitAR, VAR, FitFGN, BoxPierce, gvtest, LjungBox, Hosking, LiMcLeod, fitstable, garch, garchFit, tar.

Examples

```
#to save time the other examples are not run
## Not run:
##############################################################################
####                                                                      ####
####                  Monte-Carlo Portmanteau Tests                       ####
####                                                                      ####
##############################################################################
## Monte-Carlo test for randomness series                                  ##
##############################################################################
data("DEXCAUS")
returns <- log(DEXCAUS[-1]/DEXCAUS[-length(DEXCAUS)])
portest(returns)            ## MC using one CPU takes about 25.16 seconds
portest(returns, nslaves=4) ## MC using 4 CPUs takes about 9.51 seconds
portest(returns, MonteCarlo=FALSE)                ## asymptotic PenaRodriguez
portest(returns,test="LjungBox", MonteCarlo=FALSE) ## asymptotic LjungBox
##############################################################################
## Monte-Carlo goodness-of-fit arima test using 4 CPUs                    ##
##############################################################################
## arima() or Arima() function takes about 14.32 seconds
ans1 <- arima(WWWusage,order=c(3,1,0))
portest(ans1, nworkers = 4)
#
## arima0() function takes about 15.26 seconds
ans2 <- arima0(WWWusage,order=c(3,1,0))
portest(ans2, nworkers = 4)
#
## auto.arima() function from package forecast takes about 13.59 seconds
library("forecast")
ans3 <- auto.arima(WWWusage)
portest(ans3, nworkers = 4)
#
## ar() function takes about 9.39 seconds
ans4 <- ar(Nile,order.max=2)
portest(ans4, nworkers = 4)
#
## FitAR() function takes about 10.78 seconds
library("FitAR")
ans5 <- FitAR(Nile, p=2)
portest(ans5, nworkers = 4)
##############################################################################
## Monte-Carlo goodness-of-fit VAR test - Multivariate series     ##
##############################################################################
data("IbmSp500")
ibm <- log(IbmSp500[,2]+1)*100
sp500 <- log(IbmSp500[,3]+1)*100
IBMSP500 <- data.frame(cbind(ibm,sp500))
## ar.ols() function takes about 9.11 seconds
ans6 <-  ar.ols(IBMSP500, aic=FALSE, intercept=TRUE, order.max=5)
portest(ans6, NREP=100, test="PenaRodriguez", nworkers=4)
## VAR() function takes about 11.55 seconds
library("vars")
```

```
ans7 <- VAR(IBMSP500, p=5)
portest(ans7, NREP=100, test="PenaRodriguez", nworkers=4)
portest(ans7,test="Hosking", MonteCarlo=FALSE) ## asymptotic Hosking test
#############################################################################
## Monte-Carlo test for GARCH effects using 4 CPUs                         ##
#############################################################################
## Example 1
## Test for ARCH effects on returns series takes about 14.65 seconds
data("monthintel")
returns <- as.ts(monthintel)
lags <- c(5, 10, 20, 40)
portest(returns, lags = lags, nworkers = 4, SquaredQ = TRUE)
#
## Example 2
library("fGarch")
library("tseries")
data("GNPDEF")
z<-ts(GNPDEF[,2], start=1947, freq=4)
r <- 100*diff(log(z))
## use garch() function takes about 6.75 seconds
FitGarch1 <- garch(r, order = c(1,1))
portest(FitGarch1,NREP=100,nworkers = 4)
portest(FitGarch1,NREP=100,nworkers = 4,SquaredQ=TRUE)
#
## use garchFit() function takes about 13.56 seconds
GarchFit2 <- garchFit(formula = ~arma(4,0)+garch(1,1), data=r, trace=FALSE)
portest(GarchFit2, NREP=100, nworkers = 4, SquaredQ = FALSE)
portest(GarchFit2, NREP=100, nworkers = 4, SquaredQ = TRUE)
#############################################################################
## Monte-Carlo test on residuals with infinite variances                   ##
#############################################################################
## It takes about 32.7 seconds on personal PC with 4 CPUs
data("CRSP")
CRSP.AR5<- arima(CRSP, c(5, 0, 0))
lags <- c(10, 20, 30)
portest(CRSP.AR5,lags=lags,nworkers=4,NREP=1000,InfiniteVarianceQ=TRUE)
#############################################################################
## Monte-Carlo test for Fractional Gaussian Noise, FGN.                    ##
#############################################################################
## It takes about 55.06 seconds on personal PC with 4 CPUs
library("FGN")
data("NileMin")
NILE.FGN <- FitFGN(NileMin)
lags <- c(5, 10, 20)
portest(NILE.FGN, lags = lags, nworkers = 4)
portest(NILE.FGN, MonteCarlo=FALSE)  ## asymptotic distribution method
#################################################################
## Write two functions to fit a model and simulate results
## Apply Monte-Carlo test on fitted obj with class "list"
#################################################################
## Example 1
## Threshold Autoregressive (TAR) Model example from TSA package
## It takes about 64.27 seconds on personal PC with 4 CPUs
```

```
library("TSA")
FitModel <- function(data){
    fit <- TSA::tar(y=log(data),p1=4,p2=4,d=3,a=0.1,b=0.9,print=FALSE)
    res <- ts(fit$std.res)
    parSpec <- list(res=res,fit=fit)
  parSpec
}
SimModel <- function(parSpec){
    fit <- parSpec$fit
  exp(tar.sim(fit)$y)
}
data(prey.eq)
portest(FitModel(prey.eq),nworkers=4,func=list(SimModel,FitModel),pkg="TSA")
#
## Example 2
## It takes about 10.75 seconds on personal PC with 4 CPUs
FitModel <- function(data){
    fit <- ar(data,aic = FALSE, order.max=2)
    order <- 2
    res <- ts(fit$resid[-(1:order)])
    phi <- fit$ar
    theta <- NULL
    sigma <- fit$var.pred
    demean <- fit$x.mean
 list(res=res,phi=phi,theta=theta,sigma=sigma,demean=demean)
}
SimModel <- function(parSpec){
    res <- parSpec$res
    n <- length(res)
    innov <- sample(x=res,size=n,replace = TRUE)
    phi <- parSpec$phi
    theta <- parSpec$theta
    sigma <- parSpec$sigma
    demean <- parSpec$demean
   arima.sim(n = n, list(ar = phi, ma = theta), innov = innov,
             sd = sqrt(sigma), mean = demean)
}
Fit <- FitModel(Nile)
portest(Fit,nworkers=4,func=list(SimModel=SimModel,FitModel=FitModel),pkg="stats")

## End(Not run)
```

`rStable` *Generate Data From Stable Distributions*

Description

Generate data from stable distribution with infinite variance.

Usage

```
rStable(n, ALPHA, BETA, GAMMA = NULL, DELTA = NULL)
```

Arguments

n	length of the series.
ALPHA	index parameters, each in the range (0,2].
BETA	skewness parameters, each in the range [-1, 1].
GAMMA	scale parameters.
DELTA	location parameters.

Details

ALPHA, BETA, GAMMA, and DELTA should have the same length. This length, k, represents the number of the variables that we need to generate. The code in the function rStable extends that one given in the package fBasics to the multivariate case. Many thanks to Diethelm Wuertz for putting his code under the GPL license.

Value

A vector of dimension $n \times k$ from independent stable distributions.

Author(s)

Esam Mahdi and A.I. McLeod.

References

Chambers, J.M., Mallows, C.L., and Stuck, B.W. (1976). "A Method for Simulating Stable Random Variables". Journal of American Statistical Association, 71, 340-344.

Wuertz, D., core team members R (2012). "fBasics: Rmetrics - Markets and Basic Statistics". R package version 2160.85. http://CRAN.R-project.org/package=fBasics

See Also

There is also a function rstable in the fBasics package for the univariate case only. See also fitstable, varima.sim

Examples

```
## Generate Univariate Data
n <- 500
ALPHA <- 1.75
BETA <- 0
GAMMA <- 1.5
DELTA <- 0
rStable(n, ALPHA, BETA, GAMMA, DELTA)
#to save time the other examples are not run
## Not run:
## Generate Bivariate Data
n <- 500
ALPHA <- c(1.3,1.5)
BETA <- c(0.3,-0.6)
```

```
rStable(n, ALPHA, BETA)
## Generate Multivariate Data
n <- 500
ALPHA <- c(1.3,1.5,1.7)
BETA <- c(0.3,-0.6,0)
GAMMA <- c(3,1,6)
rStable(n, ALPHA, BETA,GAMMA)

## End(Not run)
```

`ToeplitzBlock` *Toeplitz Block Matrix of Hosking (1980) Auto and Cross Correlation Matrices*

Description

Block Toeplitz matrix of order $m + 1$ with $k \times k$ auto-cross correlation matrices. The Hosking (1980) definition of the correlation matrix is used. This is needed for the function `gvtest`.

Usage

```
ToeplitzBlock(res,lag.max,Kernel)
```

Arguments

`res`	residuals, numeric or matrix.
`lag.max`	an integer number = m is used to determined the order of the block matrix.
`Kernel`	Logic number. default = FALSE. Parzen Kernel would be used for the autocorrelation function if setting TRUE

Value

A block Toeplitz matrix of auto and cross correlation matrices using Hosking (1980) definition from `lag` = 0 to `lag` = m.

Author(s)

Esam Mahdi, Ken Jinkun Xiao and A.I. McLeod.

References

Hosking, J. R. M. (1980). "The Multivariate Portmanteau Statistic". Journal of American Statistical Association, 75, 602-608.

Lin, J.-W. and McLeod, A.I. (2006). "Improved Generalized Variance Portmanteau Test". Computational Statistics and Data Analysis, 51, 1731-1738.

Mahdi, E. and McLeod, A.I. (2011, accepted). "Improved Multivariate Portmanteau Test". Journal of Time Series Analysis. (JTSA - 3192).

See Also

acf, gvtest, toeplitz

Examples

```
## Univariate Series
x <- rnorm(100)
ToeplitzBlock(x,lag.max=4)
## Univariate Series with Parzen Kernel
x<-rnorm(100)
ToeplitzBlock(x,lag.max=4,Kernel=TRUE)

## Multivariate Series
x <- cbind(rnorm(100),rnorm(100))
ToeplitzBlock(x,lag.max=4)
```

---

varima.sim *Simulate Data From Nonseasonal ARIMA(p,d,q) or VARIMA(p,d,q) Models*

Description

Simulate time series from Integrated AutoRegressive Moving Average, ARIMA(p,d,q), or Vector Integrated AutoRegressive Moving Average, VARIMA(p,d,q), where d is a nonnegative difference integer in the ARIMA case and it is a vector of k differenced components $d_1, ..., d_k$ in the VARIMA case. The simulated process may have a deterministic terms, drift constant and time trend, with non-zero mean. The innovations may have finite or infinite variance.

Usage

```
varima.sim(phi=NULL,theta=NULL,d=NA,sigma,n,constant= NA,trend=NA,
         demean=NA,innov=NULL,innov.dist=c("normal","t","stable"),
         df=1,StableParameters=NA,Trunc.Series=NA)
```

Arguments

phi	a numeric or an array of AR or an array of VAR parameters with order p.
theta	a numeric or an array of MA or an array of VMA parameters with order q.
d	an integer or a vector representing the order of the difference.
sigma	variance of white noise series. It must be entered as matrix in case of bivariate or multivariate time series.
n	length of the series.
constant	a numeric vector represents the intercept in the deterministic equation.
trend	a numeric vector represents the slop in the deterministic equation.
demean	a numeric vector represents the mean of the series.

innov	univariate or multivariate innovation series.
innov.dist	innovation distribution from univariate or multivariate normal, t, or stable. Default is normal when innov = NULL is selected.
df	degrees of freedom needed for generating innovations with univariate or multivariate t-distribution.
StableParameters	the four parameters, ALPHA, BETA, GAMMA, and DELTA, as described in the function rStable. This argument is needed when both arguments innov.dist = "stable" and innov = NULL are selected. These parameters are used to generate data from ARIMA or VARIMA with innovations from stable distribution with infinite variance.
Trunc.Series	truncation lag is used to truncate the infinite MA or VMA Process. IF it is NA, then Trunc.Series = min(100,$n/3$).

Details

This function is used to simulate a nonseasonal ARIMA or VARIMA model of order (p, d, q)

$$\Phi(B)D(B)(Z_t - \mu) = a + b \times t + \Theta(B)e_t,$$

where $a, b,$ and μ correspond to the arguments constant, trend, and demean respectively. The univariate or multivariate series e_t represents the innovations series given from the argument innov. If innov = NULL then e_t will be generated from a univariate or multivariate normal distribution, t-distribution, or stable distribution of infinite variance specified from the argument innov.dist. $\Phi(B)$ and $\Theta(B)$ are the VAR VMA coefficient matrices respectively and B is the backshift time operator. $D(B) = diag[(1-B)^{d_1}, \ldots, (1-B)^{d_k}]$ is a diagonal matrix. This states that each individual series $Z_i, i = 1, ..., k$ is differenced d_i times to reduce to a stationary VARMA(p,q) series.

Value

Simulated data from ARIMA(p,d,q) or VARIMA(p,d,q) process that may have a drift and deterministic time trend terms.

Author(s)

Esam Mahdi and A.I. McLeod.

References

Hipel, K.W. and McLeod, A.I. (2005). "Time Series Modelling of Water Resources and Environmental Systems".

Reinsel, G. C. (1997). "Elements of Multivariate Time Series Analysis". Springer-Verlag, 2nd edition.

See Also

arima.sim, vma.sim, ImpulseVMA, InvertQ, fitstable, rStable

Examples

```
#########################################################################
# Simulate MA(1) where innovation series is provided via argument innov
#########################################################################
set.seed(1234)
n <- 200
phi <-  NULL
theta <-  0.6
d <- NA
sigma <- 1.9
Z <- varima.sim(phi, theta, d, sigma, n,innov=rnorm(n))
plot(Z)
#to save time the other examples are not run
## Not run:
#########################################################################
# Simulate ARIMA(2,1,0) process with phi=c(1.3,-0.35), Gaussian innovations
# The series is truncated at lag 50
#########################################################################
set.seed(1234)
Trunc.Series <- 40
n <- 1000
phi <- c(1.3, -0.35)
theta <- NULL
d <- 1
sigma <- 1
Z <- varima.sim(phi,theta,d,sigma,n,Trunc.Series=Trunc.Series)
coef(arima(Z,order=c(2,1,0)))
#########################################################################
# Simulate MA(1) process with theta = 0.5, t5-distribution innovations
#########################################################################
set.seed(1234)
n <- 200
phi <- NULL
theta <- 0.5
Z <- varima.sim(phi, theta, sigma=1, n=n, innov.dist="t", df=5)
plot(Z)
#########################################################################
# Simulate univariate ARMA(2,1) process with length 500,
# phi = c(1.3, -0.35), theta = 0.1. Drift equation is 8 + 0.05*t
# Stable innovations with: ALPHA = 1.75, BETA = 0, GAMMA = 1, DELTA = 0
#########################################################################
set.seed(1234)
n <- 500
phi <-  c(1.3, -0.35)
theta <-  0.1
constant <- 8
trend <- 0.05
demean <- 0
d <- 0
sigma <-  0.7
ALPHA <- 1.75
BETA <- 0
```

```
GAMMA <- 1
DELTA <- 0
Stable <- c(ALPHA,BETA,GAMMA,DELTA)
Z <- varima.sim(phi,theta,d,sigma,n,constant,trend,demean,
     innov.dist="stable",StableParameters=Stable)
plot(Z)
#########################################################################
# Simulate a bivariate white noise series from a multivariate t4-distribution
#########################################################################
set.seed(1234)
Z <- varima.sim(sigma=diag(2),n=200,innov.dist="t",df=4)
plot(Z)
#########################################################################
# Simulate a trivariate VARMA(1,1) process with length 300.
# phi = array(c(0.5,0.4,0.1,0.5,0,0.3,0,0,0.1), dim=c(k,k,1)), where k =3
# theta = array(c(0,0.25,0,0.5,0.1,0.4,0,0.25,0.6), dim=c(k,k,1)).
# innovations are generated from multivariate normal distribution
# The process have mean c(10, 0, 12),
# Drift equation a + b * t, where a = c(2,1,5), and b = c(0.01,0.06,0)
# The series is truncated at default value: Trunc.Series=ceiling(100/3)=34
#########################################################################
set.seed(1234)
k <- 3
n <- 300
Trunc.Series <-  50
phi <-  array(c(0.5,0.4,0.1,0.5,0,0.3,0,0,0.1),dim=c(k,k,1))
theta <-  array(c(0,0.25,0,0.5,0.1,0.4,0,0.25,0.6),dim=c(k,k,1))
sigma <- diag(k)
constant <- c(2,1,5)
trend <- c(0.01,0.06,0)
demean <- c(10,0,12)
Z <- varima.sim(phi, theta, d = 0,sigma, n, constant,trend,demean)
plot(Z)
#########################################################################
# Simulate a bivariate VARIMA(2,d,1) process with length 300, where d=(1,2).
# phi = array(c(0.5,0.4,0.1,0.5,0,0.3,0,0),dim=c(k,k,2)),
# theta = array(c(0,0.25,0,0), dim=c(k,k,1)).
# innovations are generated from multivariate normal
# The process have mean zero and no deterministic terms.
# The variance covariance is sigma = matrix(c(1,0.71,0.71,2),2,2).
# The series is truncated at default value: Trunc.Series=ceiling(100/3)=34
#########################################################################
set.seed(1234)
k <- 2
n <- 300
Trunc.Series <-  50
phi <-  array(c(0.5,0.4,0.1,0.5,0,0.3,0,0),dim=c(k,k,2))
theta <-  array(c(0,0.25,0,0),dim=c(k,k,1))
d <- c(1,2)
sigma <- matrix(c(1,0.71,0.71,2),k,k)
Z <- varima.sim(phi, theta, d, sigma, n)
plot(Z)
#########################################################################
```

```
# Simulate a bivariate VAR(1) process with length 600.
# Stable distribution: ALPHA=(1.3,1.6), BETA=(0,0.2), GAMMA=(1,1), DELTA=(0,0.2)
# The series is truncated at default value: Trunc.Series=min(100,200)=100
#########################################################################
set.seed(1234)
k <- 2
n <- 600
phi <- array(c(-0.2,-0.6,0.3,1.1),dim=c(k,k,1))
theta <- NULL
d <- NA
sigma <- matrix(c(1,0.71,0.71,2),k,k)
ALPHA <- c(1.3,1.6)
BETA <- c(0,0.2)
GAMMA <-c(1,1)
DELTA <-c(0,0.2)
Stable <- c(ALPHA,BETA,GAMMA,DELTA)
Z <- varima.sim(phi,theta,d,sigma,n,innov.dist="stable",StableParameters=Stable)
plot(Z)

## End(Not run)
```

`vma.sim` *Compute The Vector of Moving Average Model (VMA)*

Description

This utility function is useful to use in the function `varima.sim` and may used to compute the coefficients of moving-average or vector moving-average.

Usage

```
vma.sim(psi, a)
```

Arguments

`psi`	the impulse coefficients.
`a`	innovations

Value

Vector of length n (in the univariate case), or n matrices (in the multivariate case), where n = length(a)-length(Ψ) and $n \times k$ is the dimension of the series.

Author(s)

Esam Mahdi and A.I. McLeod.

References

Hannan, E.J. (1970). "Multiple Time Series". New York: Wiley.

Hipel, K.W. and McLeod, A.I. (2005). "Time Series Modelling of Water Resources and Environmental Systems".

See Also

convolve, varima.sim, arima.sim, ImpulseVMA, InvertQ, fitstable

Examples

```
k <- 2
n <- 300
Trunc.Series <-  50
phi <-  array(c(0.5,0.4,0.1,0.5),dim=c(k,k,1))
theta <-  array(c(0,0.25,0,0),dim=c(k,k,1))
sigma <- matrix(c(1,0.71,0.71,2),k,k)
p <- ifelse(is.null(phi),0,dim(phi)[3])
q <- ifelse(is.null(theta),0,dim(theta)[3])
r <- max(p, q)
d <- Trunc.Series + r
psi <- ImpulseVMA(phi = phi, theta = theta, Trunc.Series = Trunc.Series)
a <- t(crossprod(chol(sigma),matrix(rnorm(k*d),ncol=d)))
vma.sim(psi = psi, a = a)
```

WestGerman *Quarterly, West German Investment, Income, and Consumption: 1960Q1-1982Q4*

Description

Quarterly, seasonally adjusted, West German fixed investment, disposable income, consumption expenditures in billions of DM, 1960Q1-1982Q4.

Usage

```
data(WestGerman)
```

Format

A data frame with 92 observations on the following 3 variables.

`invest` a numeric vector denotes the investment in billions of DM

`income` a numeric vector denotes the income in billions of DM

`cons` a numeric vector denotes the consumption expenditures in billions of DM

Source

Deutsche Bundesbank; http://www.jmulti.de/data_imtsa.html

References

Lutkepohl, H. (2005). "New introduction to multiple time series analysis". Springer-Verlag, New York.

Index