Package ‘marima’

January 14, 2017

Type Package
Title Multivariate ARIMA and ARIMA-X Analysis
Version 2.2
Date 2017-01-13
Author Henrik Spliid
Maintainer Henrik Spliid <hsp1@dtu.dk>
Description Multivariate ARIMA and ARIMA-X estimation using Spliid's algorithm (marima()) and simulation (marima.sim()).
License GPL-2
RoxygenNote 5.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2017-01-14 00:59:20

R topics documented:

arma.filter .................................................. 2
arma.forecast ............................................... 3
austr .......................................................... 5
C20 ............................................................ 6
check.one .................................................... 8
define.dif ................................................... 9
define.model ................................................ 10
define.sum ................................................... 13
forec.var ..................................................... 13
inverse.form ................................................. 14
lead.one ....................................................... 15
marima ......................................................... 15
marima.sim .................................................. 19
pol.inv ......................................................... 20
pol.mul ....................................................... 21
pol.order ..................................................... 22
Description

Filtering of (kvar-variate) time series with marima type model.

Calculation of residuals and filtered values of timeseries using a marima model.

Usage

arma.filter(series = NULL, ar.poly = array(diag(kvar), dim = c(kvar, kvar, 1)), ma.poly = array(diag(kvar), dim = c(kvar, kvar, 1)), means = 1)

Arguments

series matrix holding the kvar by n multivariate timeseries (if (kvar > n) the series is transposed and a warning is given).

ar.poly (kvar, kvar, p+1) array containing autoregressive matrix polynomial model part.
If the filtering is to be performed for undifferenced data when the analysis (in marima) was done for differenced data, the input array ar.poly should incorporate the ar-representation of the differencing operation (using, for example: ar.poly <- pol.mul(ar.estimate, dif.poly, L = (dim(ar.estimates)[3]+dim(dif.poly)[3])), where 'dif.poly' was obtained when differencing the time series (using define.dif) before analysing it with marima (giving the ar.estimate).

ma.poly (kvar, kvar, q+1) array containing moving average matrix polynomial model part.
If a leading unity matrix is not included in the ar- and/or the ma-part of the model this is automatically taken care of in the function (in that case the dimensions of the model arrays used in arma.filter() are, respectively, (kvar, kvar, p+1) and (kvar, kvar, q+1)).

means vector (length = kvar) indicating whether means are subtracted or not (0/1). Default : means = 1 saying that all means are subtracted (equivalent to means = c(1, 1, ..., 1)).
Value

estimates = estimated values for input series
residuals = corresponding residuals. It is noted that the residuals computed by arma.filter may
deviate slightly from the marima-residuals (which are taken from the last repeated regression step
performed). The residuals computed by arma.filter are constructed by filtering (successive use of
the arma model) and using a heuristic method for the first residuals.
averages = averages of variables in input series
mean.pattern = pattern of means as used in filtering

Examples

```
library(marima1)
data(austr)
series<-t(austr)[,1:90]
# Define marima model
Model5 <- define.model(kvar=7, ar=1, ma=1, rem.var=1, reg.var=6:7)

# Estimate marima model
Marima5 <- marima(series, Model5$ar.pattern, Model5$ma.pattern, penalty=1)

# Calculate residuals by filtering
Resid <- arma.filter(series, Marima5$ar.estimates, Marima5$ma.estimates)

# Compare residuals
plot(Marima5$residuals[2, 5:90], Resid$residuals[2, 5:90], xlab='marima residuals', ylab='arma.filter residuals')
```

Description

Forecasting of (multivariate) time series of using marima type model.

Usage

```
arma.forecast(series = NULL, marima = NULL, nstart = NULL, nstep = 1,
dif.poly = NULL, check = TRUE)
```
Arguments

series matrix holding the kvar-variate timeseries. The series is assumed to have the same format as the timeseries analysed by marima BEFORE differencing (if differencing was used via define.dif) (the length, though, does not need to be the same but can be shorter or longer). Results from estimating the model (for the differenced data, if used) are assumed to be saved in the input-object 'marima' (see 'usage') by marima.

The series is assumed to have the total length=(nstart+nstep) (but it may be longer. In any case the forecasting is starting from nstart continuing to nstart+nstep. Future values already present or initialised, for example, as NAs are overwritten with the forecasted values.)

An example of a series prepared for forecasting is in the marima library: 'data(austr)'; (see below, the example).

If future (independent) x-values for the forecasting are to be used these values must be supplied in 'series' at the proper places before calling 'arma.forecast(...)' (that is except the x-value(s) corresponding to the last prediction).

marima the object holding the marima results to be used for the forecasting, that is an output object created by marima.

If the ar- and/or the ma-model do not include a leading unity matrix this is automatically taken care of in the function (in that case the dimensions of the model arrays used will be, respectively, (kvar, kvar, p+1) and (kvar, kvar, q+1)) after inserting the leading unity matrix (if the object 'marima' was produced by marima, this will automatically be OK.

nstart starting point for forecasting (1st forecast values will be for time point t = nstart+1).

nstep length of forecast (forecasts will be for time points nstart+1,...,nstart+nstep).

dif.poly (most often) output from the function define.dif holding the ar-representation of the differencing polynomial (define.dif$dif.poly). If a differenced timeseries was analysed by marima the forecast-variance/covariance matrices are calculated for the aggregated (original) timeseries if 'dif.poly' is specified. If not, the forecast-variance/covariance matrices are calculated for the differenced time series. If forecasting is wanted for the original (not differenced) time series the 'dif.poly' created by define.dif must be specified.

check If check=TRUE (default) various checks and printouts are carried out.

Value

forecasts = forecasted values following the nstart first values of the input series (at time points 'nstart+1,...,nstart+nstep'). The forecasted values will be (over-) written in the input series at the proper future positions (if relevant).

residuals = corresponding residuals for input series followed by nstep future residuals (all=0).

prediction.variances = (kvar, kvar, nstep) array containing prediction covariance matrices corresponding to the nstep forecasts.

nstart = starting point for prediction (1st prediction at point nstart+1).

nstep = length of forecast
Examples

```r
library(marima)
data(austr)
series<-austr
Model5 <- define.model(kvar=7, ar=1, ma=1, rem.var=1, reg.var=6:7)
Marima5 <- marima(ts(series[1:90,]), Model5$ar.pattern, Model5$ma.pattern, penalty=1)

nstart <- 90
nstep <- 10
cat("Calling arma.forecast.\n")
cat("In the example the input series is dim(length,kvar).\n")
cat("and of type ts() (timeseries) for illustration. \n")
Forecasts <- arma.forecast(series=ts(series), marima=Marima5, nstart=nstart, nstep=nstep )
Year<-series[91:100,1]
One.step <- Forecasts$forecasts[, (nstart+1)]
One.step
Predict <- Forecasts$forecasts[ 2, 91:100]
Predict
stdv<-sqrt(Forecasts$pred.var[2, 2, ])
upper.lim=Predict+stdv*1.645
lower.lim=Predict-stdv*1.645
Out<-rbind(Year, Predict, upper.lim, lower.lim)
print(Out)
# plot results:
plot(series[1:100, 1], Forecasts$forecasts[2, ], type='l', xlab='Year',
ylab='Rate of armed suicides', main='Prediction of suicides by firearms',
ylim=c(0.0, 4.1))
lines(series[1:90, 1], series[1:90, 2], type='p')
grid(lty=2, lwd=1, col='black')
Years<-2005:2014
lines(Years, Predict, type='l')
lines(Years, upper.lim, type='l')
lines(Years, lower.lim, type='l')
lines(c(2004.5, 2004.5), c(0.0, 2.0), lty = 2)
```

**austr**  
*Data set for testing marima package (australian killings)*

**Description**  
Data for marima examples.

**Usage**  
```r
data(austr)
```
Format

A data frame (austr) with 7 columns and 100 rows.

Year  Year for data
suic.fire  Rate of suicides by firearms
homi.fire  Rate of homicides by firearms
suic.other  Rate of suicides by non firearms
homi.other  Rate of homicides by non firearms
leg  Legislation against firearms in effect
acc.elg  Accumulated effect of legislation in years

| C20 | Data set for testing marima package (Copenhagen Stocks) |

Description

Two years of prices for 18 shares from the Copenhagen Stock Exchange C20 index, covering the most valuable companies. Two shares have been removed (Maersk A = almost identical to Maersk B) and ISS which is incomplete for the period considered.

Usage

data(C20)

Format

A data frame (C20) with 1+18+18 columns and 517 rows (about two full years).

Dates  Format for date is 2016-04-01
CARL.fin  Closing price for stock 'Carlsberg'.
CHR..fin  Closing price for stock 'Christian Hansen'.
COLO.fin  Closing price for stock 'Coloplast'.
DANS.fin  Closing price for stock 'Danske Bank'.
DSV..fin  Closing price for stock 'DSV'.
GEN..fin  Closing price for stock 'Genmap'.
GN.2.fin  Closing price for stock 'GN St. Nord'.
FLS..fin  Closing price for stock 'FL Smidth'.
JYSK.fin  Closing price for stock 'Jyske Bank'.
MAER.fin  Closing price for stock 'Maersk B'.
NDA..fin  Closing price for stock 'Nordea Bank'.
NOVO.fin  Closing price for stock 'Novo'.
NZYM.fin Closing price for stock 'Novozymes'.
PDNO.fin Closing price for stock 'Pandora'.
TDC..fin Closing price for stock 'TDC'.
TRYG.fin Closing price for stock 'Pandora'.
VWS..fin Closing price for stock 'Vestas Wind'.
WDH..fin Closing price for stock 'William Demant'.
CARL.ave Average price for stock 'Carlsberg'.
CHR..ave Average price for stock 'Christian Hansen'.
COLO.ave Average price for stock 'Coloplast'.
DANS.ave Average price for stock 'Danske Bank'.
DSV..ave Average price for stock 'DSV'.
GEN..ave Average price for stock 'Genmap'.
GN.2.ave Average price for stock 'GN St. Nord'.
FLS..ave Average price for stock 'FL Smidth'.
JYSK.ave Average price for stock 'Jyske Bank'.
MAER.ave Average price for stock 'Maersk B'.
NDA..ave Average price for stock 'Nordea Bank'.
NOVO.ave Average price for stock 'Novo'.
NZYM.ave Average price for stock 'Novozymes'.
PDNO.ave Average price for stock 'Pandora'.
TDC..ave Average price for stock 'TDC'.
TRYG.ave Average price for stock 'Pandora'.
VWS..ave Average price for stock 'Vestas Wind'.
WDH..ave Average price for stock 'William Demant'.

Examples

# Example 1:

library(marima)
data(C20)

selects <- c(2,7,11)

cat("Multivariate model for ",colnames(C20)[selects]," \n")

Data <- data.frame(C20[,selects])
colnames(Data) <- colnames(C20)[selects]

log.Data <- log(Data)
kvar <- length(selects)
k <- c(1:kvar)
```r
difs <- rep(1, length(selects))
difference <- rbind(k, difs)
dlog.Data <- 100*t(define.dif(log.Data, difference)$y.dif)
cat("dlog.Data represents the percentage change from day to day. \n")
mod <- define.model(kvar = kvar, ar=c(1:2), ma=c(1))
Model <- marima(dlog.Data,
    ar.pattern=mod$ar.pattern, ma.pattern=mod$ma.pattern, penalty=2)
short.form(Model$ar.estimates, leading=FALSE)
short.form(Model$ma.estimates, leading=FALSE)
# Example 2:
library(marima)
data(C20)
selects <- c(13)
cat("Univariate model for ", colnames(C20)[selects]," \n")
Data <- data.frame(C20[,selects])
colnames(Data) <- colnames(C20)[selects]
log.Data <- log(Data)
kvar <- length(selects)
k <- c(1:kvar)
difs <- rep(1, length(selects))
difference <- rbind(k, difs)
dlog.Data <- 100*t(define.dif(log.Data, difference)$y.dif)
mod <- define.model(kvar = kvar, ar=c(1:2), ma=c(1))
Model <- marima(dlog.Data,
    ar.pattern=mod$ar.pattern, ma.pattern=mod$ma.pattern, penalty=2)
short.form(Model$ar.estimates, leading=FALSE)
short.form(Model$ma.estimates, leading=FALSE)
```

### Description

Function to check and insert leading unity matrix if NOT present.
**define.dif**

**Usage**
```
check.one(polyn = NULL)
```

**Arguments**
`polyn` 
(k, k, ...) matrix polynomium with or without leading unity matrix.

**Value**
`polyn` (array) with a leading unity matrix being inserted if not present.

**Examples**
```
set.seed(4711)
X <- array(rnorm(32),dim=c(4, 4, 2))
X <- check.one(X)
short.form(X)
```

---

**Description**
Function to generate and apply a differencing matrix polynomial (autoregressive form) defined by a pattern.
To be used before calling marima in order to difference the timeseries before the marima analysis. The averages of the variables in the time series are subtracted from the input series before differencing.

**Usage**
```
define.dif(series = series, difference = NULL)
```

**Arguments**
`series` 
k-variate timeseries (kvar by n matrix).
`difference` 
2 by L matrix defining L differencing operations.

**Value**
`y.dif` = the differenced timeseries (the complete part)
`y.lost` = the first observations lost because of differencing
`dif.poly` = differencing polynomial array = c(kvar, kvar, ...) holding the autoregressive representation of the specified differencing
`averages` = the averages of the original series as they were subtracted before differencing
`dif.series` = the differenced series (y.lost followed by y.dif)
Examples

# Generate Y=series with 4 variables for illustration:
set.seed(4711)
Y<-matrix(round(100*rnorm(40)+10), nrow=4)

# Example 1: use of difference parameter: If
difference=c(2, 1, 2, 1, 3, 12)
difference
# the variable 2 is differenced
# twice, and variable 3 is differenced once with lag=12.

# Example 2:
poly <- define.dif(series=Y, difference=c(2, 1, 3, 1, 3, 1))
poly
# Generates a (4-variate) polynomial differencing array (with a leading
# unity matrix corresponding to lag=0, and (in the example) differencing
# of variable 2 for lag 1 and variable 3 for lag 1 but twice. Afterwards
# the series Y is differenced accordingly. Results in poly$series and
# poly$dif.poly.

# Example 3: Generation and application of multivariate differencing
# polynomial. Re-use the 4-variate time series and use the
# differencing polynomial (ar-form):
# var=1, dif=1, var=2, dif=6, and var=3 and 4, no differencing.
dif.y <- define.dif(Y, c(1, 1, 2, 6, 3, 0, 4, 0))
# Now dif.y contains the differenced series and the differencing
# polynomial. Print the generated polynomial in short form:
short.form(dif.y$dif.poly)
# Specifying no differencing (3, 0 and 4, 0) may be omitted:
dif.y <- define.dif(Y, c(1, 1, 2, 6))
dif.y

# Example 4:
y<-matrix(round(rnorm(1200)*100+50), nrow=6)
library(marima)
difference<-c(3, 2, 4, 0, 5, 0, 6, 7)
matrix(difference, nrow=2)
Y<-define.dif(y, difference=difference)
round(rowMeans(Y$dif.series), 2)
round(Y$averages, 2)

define.model

Description

Function to define multivariate arma model (indicator form) for marima.
Usage

define.model(kvar = 1, ar = 0, ma = 0, rem.var = 0, reg.var = 0,
    no.dep = NULL, print = 0, ar.fill = NULL, ar.rem = NULL,
    ma.fill = NULL, ma.rem = NULL, indep = NULL)

Arguments

kvar  
dimension of time series

ar  
autoregression definition. For example ar=c(1, 2, 12) will generate autoregression at lags 1, 2 and 12.

ma  
moving average definition. Works like ar. If ma=c(1, 2) moving average terms at lags 1 and 2 are defined.

rem.var  
o. of variable(s) not to be considered in marima.

reg.var  
o. of variable(s) that can only act as regression variable(s) such as (typically) a so-called leading indicator.

no.dep  
sequence of pairs of variables. For example no.dep=c(1, 2, 2, 3) means that variable 2 is not allowed in model for variable 1, and variable 3 is not allowed in model for variable 2.

print  
(!0/0) If !0 is used, the generated patterns of the arma model and other informations are printed on the console. If 0 is used, no printout of the arma patterns are given.

ar.fill  
sequence of triplets: c(dependent variable, independent variable, lag). ar.fill=c(2, 3, 12): Insert ar-indicator for model for dependent variable 2 and independent variable 3 at lag 12.

ar.rem  
sequence of triplets c(dependent variable, independent variable, lag). ar.rem=c(2, 3, 12): remove (if present) ar-indicator for model for dependent variable 2 and independent variable 3 at lag 12.

ma.fill  
sequence of triplets: c(dependent variable, independent variable, lag). ma.fill=c(2, 3, 12): Insert ma-indicator for model for dependent variable 2 and independent variable 3 at lag 12.

ma.rem  
sequence of triplets c(dependent variable, independent variable, lag). ma.rem=c(2, 3, 12): remove (if present) ma-indicator for model for dependent variable 2 and independent variable 3 at lag 12.

The various parameters may (in some cases) accomplish the same model requirements. The routine define.model apply these input parameters successively in the following order: 1) rem.var, 2) reg.var, 3) indep, 4) no.dep, 5) ar.fill, 6) ar.rem, 7) ma.fill, 8) ma.rem

The parameters ar.fill, ar.rem, ma.fill and ma.rem are applied last, and in that order. They overwrite what previously has been defined.

indep  
o. of variable(s) that are independent of the other variables. indep(2, 4) makes variables 2 and 4 independent of all other variables. Variables 2 and 4 may influence other variables.
Value

ar.pattern a matrix polynomial (an array) with 1’s and 0’s defining the autoregressive matrix polynomial to be fitted by marima (an array with dim=c(kvar, kvar, 1+ar_order) (with leading unity matrix)).

ma.pattern a matrix polynomial (an array) with 1’s and 0’s defining the moving average matrix polynomial to be fitted by marima (an array with dim=c(kvar, kvar, 1+ma_order) (including the leading unity matrix)).

Examples

# Example 1: 3-variate arma model with ar-lags at 1 and 2, and an
# ma-term at lag 1. And var=3 is a regression variable (X-variable).
# Model1 <- define.model(kvar=3, ar=c(1, 2), ma=c(1), reg.var=3)
# short.form(Model1$ar.pattern)
# short.form(Model1$ma.pattern, leading=FALSE)
# The object Model1 contains the ar- and ma-pattern arrays as defined.
# Model1$ar.pattern and Model1$ma.pattern are used as input to
# marima in order to define the model to be estimated.

# Example 2: arma model with ar-lags at 1, 2 and 6, and var=3
# regression variable (X-variable).
# Model2 <- define.model(kvar=3, ar=c(1, 2, 6), ma=c(1), reg.var=3)
# Print the ar- and ma-polynomial patterns using
# short.form(Model2$ar.pattern, leading=FALSE)
# short.form(Model2$ma.pattern, leading=TRUE)

# Example 3: arma model with ar-lags at 1, 2 and 6, and reg.var=3
# (X-variable). ma-order=1. Finally (ar.fill=c(2, 3, 4) puts a '1'
# for (dep-var=2, indep-var=3, ar-lag=4).
# If further modifications of the ar- or ma-patterns are needed, it
# can be accomplished before calling marima (Model3$ar.pattern and
# Model3$ma.pattern are arrays).
# Model3 <- define.model(kvar=3, ar=c(1, 2, 6), ma=c(1), reg.var=3,
# ar.fill=c(2, 3, 4))
# short.form(Model3$ar.pattern)
# short.form(Model3$ma.pattern)

# Model4 <- define.model(kvar=3, ar=c(1, 2, 6), ma=c(1), reg.var=3,
# ar.fill=c(2, 3, 4), indep=c(1))
# short.form(Model4$ar.pattern)
# short.form(Model4$ma.pattern, leading=FALSE)
**define.sum**

**Description**

Function to aggregate multivariate time series. Reverse of function 'define.dif'.

**Usage**

define.sum(series = NULL, difference = NULL, averages = 0)

**Arguments**

- **series**
  - series to be summed up.
- **difference**
  - differencing pattern (see define.dif).
- **averages**
  - of the individual series that (usually) have been subtracted when differencing the time series (if so, the averages are supplied in the output from define.dif(...).

**Value**

- sum.series the summed series.

**Examples**

```r
generate data
y <- round(matrix(100*rnorm(48), nrow=4))
difference <- matrix(c(1, 1, 1, 2, 1, 3, 6), nrow=2)
dy <- define.dif(y, difference)$dif.series
averages <- define.dif(y, difference)$averages
sum.y <- define.sum(dy, difference, averages)$series.sum

# Calculating forecast variances
forec.var(marima, nstep = 1, dif.poly = NULL)
```

**forec.var**

**Description**

Function for calculation of variances of nstep forecasts using a marima type model.

**Usage**

forec.var(marima, nstep = 1, dif.poly = NULL)
Arguments

marima  marima object (cov.u and ar.estimates and ma.estimates are used)
nstep  length of forecast
dif.poly  autoregressive representation of differencing polynomial as constructed by the function define.dif(...) when the time series is differenced (if so) before being analysed by marima.

Value

pred.var = variance-covariances for nstep forecasts (an array with dimension (kvar, kvar, nstep).
rand.shock = corresponding random shock representation of the model used.

Description

Calculation of inverse form for arma model

Usage

inverse.form(ar.poly, ma.poly, L)

Arguments

ar.poly  = autoregressive matrix part of model (array(k, k, ar-order)).
ma.poly  = moving average matrix part of model (array(k, k, ma-order)).
L  = order of return polynomial (length=L+1 including leading unity matrix).

Value

inverse form for arma model up to order L (array(k, k, L+1)).

Examples

set.seed(4711)
p1  <- check.one(matrix(rnorm(16), nrow=4))
p2  <- check.one(array(rnorm(32), dim=c(4, 4, 2)))
inverse  <- inverse.form(ar.poly=p1, ma.poly=p2, L=6)
short.form(inverse)
lead.one

Description

Function to add (or remove) a leading unity matrix to (from) an array (being an array representation of ar- or ma-polynomial).

Usage

lead.one(polyn = NULL, add = 0)

Arguments

polyn an input polynomium (an array).

add indicator for adding or removing unity matrix: +1 = add leading unity matrix, -1 = remove leading matrix.

Value

changed array (with leading unity matrix inserted or removed).

marima

Description

Estimate multivariate arima and arima-x models. Setting up the proper model for (especially) arima-x estimation can be accomplished using the routine 'define.model' that can assist in setting up the necessary autoregressive and moving average patterns used as input to 'marima'.

A more elaborate description of 'marima' and how it is used can be downloaded from:

http://www.imm.dtu.dk/~hspl/marima.use.pdf

Usage

marima(DATA = NULL, ar.pattern = NULL, ma.pattern = NULL, means = 1, max.iter = 50, penalty = 0, weight = 0.33, Plot = "none", Check = FALSE)
Arguments

DATA
time series matrix, dim(DATA) = c(kvar, n), where 'kvar' is the dimension of the time series and 'n' is the length of the series. If DATA is organized (n, kvar) (as a data.frame e.g.) it is automatically transposed in marima, and the user need not care about it. Also, and consequently, the output residuals and fitted values matrices are both organised c(kvar, n) at return from marima. The DATA is checked for completeness. Cases which include 'NA's or 'NaN's are initially left out. A message is given (on the console) and the active cases are given in the output object (...$used.cases). If DATA is a time series object it is transformed to a matrix and a warning is given ( if(is.ts(DATA)) DATA <- as.matrix(data.frame(DATA)) and a message is given (on the console).

ar.pattern
autoregressive pattern for model (see define.model). If ar.pattern is not specified a pure ma-model is estimated.

ma.pattern
moving average pattern for model (see define.model). If ma.pattern is not specified a pure ar-model is estimated. In this case the estimation is carried out by regression analysis in a few steps.

means
0/1 indicator vector of length kvar, indicating which variables in the analysis should be means adjusted or not. Default: means=1 and all variables are means adjusted. If means=0 is used, no variables are means adjusted.

max.iter
max. number of iterations in estimation (max.iter=50 is default which, generally, is more than enough).

penalty
parameter used in the R function 'step' for stepwise model reduction. If penalty=2, the conventional AIC criterion is used. If penalty=0, no stepwise reduction of model is performed. Generally 0<=penalty<=2 works well (especially penalty=1). The level of significance of the individual parameter estimates in the final model can be checked by considering the (approximate) 'ar.pvalues' and the 'ma.pvalues' calculated by marima.

weight
weighting factor for smoothing the repeated estimation procedure. Default is weight=0.33 which often works well. If weight>0.33 (e.g. weight=0.66) is specified more damping will result. If a large damping factor is used, the successive estimations are more cautious, and a slower (but safer) convergence (if possible) may result (max.iter may have to be increased to, say, max.iter=75.

Plot
'none' or 'trace' or 'log.det' indicates a plot that shows how the residual covariance matrix (resid.cov) develops with the iterations. If Plot= 'none' no plot is generated. If Plot= 'trace' a plot of the trace of the residual covariance matrix versus iterations is generated. If Plot='log.det' the log(determinant) of the residual covariance matrix (resid.cov) is generated. Default is Plot= 'none'.

Check
(TRUE/FALSE) results (if TRUE) in a printout of some controls of the call to arima. Useful in the first attempt(s) to use marima. Default=FALSE.

Value

Object of class marima containing:

N = N length of analysed series
kvar = dimension of time series (all random and non-random variables).
ar.estimates = ar-estimates
ma.estimates = ma-estimates
ar.fvalues = ar-fvalues (approximate)
ma.fvalues = ma-fvalues (approximate)
ar.stdv = standard deviations of ar-estimates (approximate)
ma.stdv = standard deviations of ma-estimates (approximate)
ar.pvalues = ar.estimate p-values (approximate). If in the input data two series are identical or one (or more) series is (are) linearly dependent of the the other series the routine lm(...) generates "NA" for estimates t-values, p-values and and parameter standard deviations. In marima the corresponding estimates, F-values and parameter standard deviations are set to 0 (zero) while the p-value(s) are set to "NaN". Can happen only for ar-parameters.
ma.pvalues = ma.estimate p-values (approximate)
residuals = estimated residuals (for used.cases), leading values (not estimated values) are put equal to NA
fitted = estimated/fitted values for all data (including non random variables) (for used.cases), leading values (not estimated values) are put equal to NA
resid.cov = covariance matrix of residuals (including non random variables) (computed for used.cases)
data.cov = covariance matrix of (all) input data (for used.cases)
averages = averages of input variables
Constant = estimated model constant = (sum_i(ar[, , i])) x averages
call.ar.pattern = calling ar.pattern
call.ma.pattern = calling ma.pattern
out.ar.pattern = resulting ar.pattern (after possible model reduction)
out.ma.pattern = resulting ar.pattern (after possible model reduction)
max.iter = max no. of iterations in call
penalty = factor used in AIC model reduction, if penalty=0, no AIC model reduction is performed (default).
weight = weighting of successive residuals updating (default=0.33)
used.cases = cases in input which are analysed
trace = trace(random part of resid.cov)
log.det = log(det(random part of resid.cov))
randoms = which are random variables in problem?
one.step = one step ahead prediction (for time = N+1) based on whole series from obs. 1 to N. The computation is based on the marima residuals (as taken from the last regression step in the repeated pseudo-regression algorithm).

Source

The code is an R code which is based on the article (below) by Spliid (1983). A repeated (socalled) pseudo regression procedure is used in order to estimate the multivariate arma model.
References


Spliid, H.: Estimation of Multivariate Time Series with Regression Variables:
http://www.imm.dtu.dk/~hspl/marima.use.pdf
www.itl.nist.gov/div898/handbook/pmc/section4/pmc45.htm

Examples

# Example 1:
library(marima)
# Generate a 4-variate time series (in this example):
#
kvar<-4 ; set.seed(4711)
y4<-matrix(round(100*rnorm(4*1000, mean=2.0)), nrow=kvar)
# If wanted define differencing of variable 4 (lag=1)
# and variable 3 (lag=6), for example:
y4.dif<-define.dif(y4, difference=c(4, 1, 3, 6))
# The differenced series will be in y4.dif$y.dif, the observations
# lost by differencing being excluded.
#
y4.dif$analysis<-y4.dif$y.dif
# Give lags the be included in ar- and ma-parts of model:
#
ar<-c(1, 2, 4)
ma<-c(1)
# Define the multivariate arma model using 'define.model' procedure.
# Output from 'define.model' will be the patterns of the ar- and ma-
# parts of the model specified.
#
Mod <- define.model(kvar=4, ar=ar, ma=ma, reg.var=3)
arp<-Mod$ar.pattern
map<-Mod$ma.pattern
# Print out model in 'short form':
#
short.form(arp)
short.form(map)
# Now call marima:
Model <- marima(y4.dif$analysis, ar.pattern=arp, ma.pattern=map,
## Description

Simulation of multivariate arma model of type 'marima'.

## Usage

marima.sim(kvar = 1, ar.model = NULL, ar.dif = NULL, ma.model = NULL, averages = rep(0, kvar), resid.cov = diag(kvar), seed = NULL, nstart = 0, nsim = 0)

## Arguments

- **kvar**: dimension of one observation (from kvar-variate time series).
- **ar.model**: array holding the autoregressive part of model, organised as in the marima$ar.estimates. May be empty (default = NULL) when there is no autoregressive part.
- **ar.dif**: array holding differencing polynomial of model, typically generated by applying the function define.dif. May be empty (default = NULL) when differencing is not included.
- **ma.model**: array holding the moving average part of model, organised as in the marima$ma.estimates. May be empty (default = NULL) when there is no moving average part.
- **averages**: vector holding the kvar averages of the variables in the simulated series.
- **resid.cov**: (kvar x kvar) innovation covariance matrix.
- **seed**: seed for random number generator (set.seed(seed)). If the seed is set by the user, the random number generator is initialised. If seed is not set no initialisation is done.
- **nstart**: number of extra observations in the start of the simulated series to be left out before returning. If nstart=0 in calling marima.sim a suitable value is computed (see code).
- **nsim**: length of (final) simulated series.

```r
penalty=0.0)
# The estimated model is in the object 'Model':
#
ar.model<-Model$ar.estimates
ma.model<-Model$ma.estimates
dif.poly<-y4.dif$dif.poly  # = difference polynomial in ar-form.
# Multiply the estimated ar-polynomial with difference polynomial
# to compute the aggregated ar-part of the arma model:
#
ar.aggregated <- pol.mul(ar.model, dif.poly, L=12)
# and print everything out in 'short form':
#
short.form(ar.aggregated, leading=FALSE)
short.form(ma.model, leading=FALSE)
```
Value

Simulated kvar variate time series of length = nsim.

Examples

library(marima)
data(austr)
old.data <- t(austr[, 1:83])
Model2 <- define.model(kvar=7, ar=c(1), ma=c(1),
rem.var=c(1, 6, 7), indep=NULL)
Marima2 <- marima(old.data, means=1, ar.pattern=Model2$ar.pattern,
ma.pattern=Model2$ma.pattern, Check=FALSE, Plot="none", penalty=4)

resid.cov <- Marima2$resid.cov
averages <- Marima2$averages
ar <- Marima2$ar.estimates
ma <- Marima2$ma.estimates

N <- 1000
kvar <- 7
y.sim <- marima.sim(kvar = kvar, ar.model = ar, ma.model = ma,
seed = 4711, averages = averages, resid.cov = resid.cov, nsim = N)

# Now simulate from model identified by marima (model=Marima2).
# The relevant ar and ma patterns are saved in
# Marima2$out.ar.pattern and Marima2$out.ma.pattern, respectively:
Marima.sim <- marima( t(y.sim), means=1,
ar.pattern=Marima2$out.ar.pattern,
ma.pattern=Marima2$out.ma.pattern,
Check=FALSE, Plot="none", penalty=0)

cat("Comparison of simulation model and estimates",
"from simulated data. \n")
round(Marima2$ar.estimates[, , 2], 4)
round(Marima2$ma.estimates[, , 2], 4)

round(Marima.sim$ar.estimates[, , 2], 4)
round(Marima.sim$ma.estimates[, , 2], 4)

Description

Calculation of left inverse of matrix polynomial. The leading term is expected to be the (k by k)
identity matrix. This is checked and the proper leading unity term is taken into account when the
inverse is calculated.
\textit{pol.mul}

\begin{verbatim}
phi = matrix polynomial coefficients = I, phi1, phi2, ..., phi(p).
dim(phi) = c(k, k, p+1) where k = dimension of coefficient matrices (k by k), and L = order of polynomial (length = 1+L , including the leading unity matrix).

Usage

pol.inv(phi, L)

Arguments

phi polynomium (an array) to invert
L order of inverse polynomium

Value

left inverse of phi of order L (L+1 terms including leading unity matrix)

Examples

set.seed(4711)
eta2<check.eta(array(rnorm(32),dim=c(4,4,2)))
eta2<pol.inv(eta2,L=12)
short.form(pi2)
\end{verbatim}

\textit{pol.mul}

\begin{verbatim}
Description

Calculation of product of two matrix polynomials (arrays).
If one or both leading unity matrices (of eta and theta) are missing, they are (it is) generated (and taken into account).

Usage

pol.mul(eta, theta, L)

Arguments

eta first matrix polynomial
theta second matrix polynomial
L order of output polynomial (length = L+1)

Value

matrix polynomial product af eta and theta
\end{verbatim}
Examples

```r
set.seed(4711)
p1 <- check.one(matrix(rnorm(16), nrow=4))
p2 <- check.one(array(rnorm(32), dim=c(4, 4, 2)))
p12 <- pol.mul(p1, p2, L=(2+3))
short.form(p12)
```

Description

Function to evaluate (significant) order of matrix polynomium.

Usage

```r
pol.order(polyn = NULL, digits = 12)
```

Arguments

- `polyn`: the polynomium the order of which is determined.
- `digits`: number of significant digits to be considered (values smaller than $10^\text{-digits}$ are taken to be 0 (zero)).

Value

`pol.order` order of polynomium `polyn`. (exclusive the leading unity matrix if present. `pol.order`=0 corresponds to the k by k unity matrix)

Examples

```r
pol <- array(1e-8*rnorm(96), dim=c(4,4,6))
pol[, , 1:3] <- array(rnorm(48), dim=c(4,4,3))
pol.order(polyn=pol, digits=12)
pol.order(polyn=pol, digits=4)
```
**print.marima**

**Description**

Print some (most relevant) content of a marima object.

**Usage**

```r
## S3 method for class 'marima'
print(x, estimates = TRUE, pvalues = FALSE, pattern = TRUE, fvalues = TRUE, ...)
```

**Arguments**

- `x` = a marima object with results of marima analysis.
- `estimates` = TRUE/FALSE: printout of parameter estimates.
- `pvalues` = TRUE/FALSE: printout of (approximate) p-values for parameter estimates.
- `pattern` = TRUE/FALSE: printout of model definition pattern(s).
- `fvalues` = TRUE/FALSE: printout of parameter (approximate) F-values.
- `...` = Not used.

**rand.shock**

**Description**

Calculation of random shock form for arma model

**Usage**

```r
rand.shock(ar.poly, ma.poly, L)
```

**Arguments**

- `ar.poly` = autoregressive matrix part of model
- `ma.poly` = moving average matrix part of model
- `L` = order of return polynomial (length=L+1 including leading unity matrix)

**Value**

random shock form of arma model up to order L (array(k,k,L+1))
Examples

season.lagging

set.seed(4711)
p1 <- check.one(matrix(rnorm(16), nrow=4))
p2 <- check.one(array(rnorm(32), dim=c(4, 4, 2)))
randshock <- rand.shock(ar.poly=p1, ma.poly=p2, L=6)
short.form(randshock)

Description

function which generates a matrix from summary(Model), where 'Model' is an 'lm' object. Is used by marima, in case there can be one or more non-identifiable (ar-)parameters when estimating the lm-object.

Usage

Results(Model = NULL)

Arguments

Model = an 'lm' object (with NA's for non-identifiable ar.parameters in the 'lm' specification, if so).

Value

Results = matrix with 4 columns containing the same information as summary(Model), but with "NA" rows replaced by rows = c(0, 0, 0, NaN).

season.lagging

Description

Generate new time series with (seasonally) lagged variables from lagging pattern.

Usage

season.lagging(y, lagging = NULL)
**Arguments**

- **y** = data series
- **lagging** = lagging array array describing what to be added to y: c(1, 3, 6) adds a new y3, using y1 lagged 6 time steps. lagging<-matrix(c(1, 3, 6-1, 2, 4, 12-1), nrow=3) adds two new variables (y3 and y4) using y1 lagged 6-1 time steps and y2 lagged 12-1 time steps.

**Value**

- **y.lagged** = the part of the new series (including new lagged variables) that can be entered into marima
- **y.future** = the part of the new series (including new lagged variables) that does not include future observation
- **y.lost** = previous values of the time series that is incomplete with respect to the new variables generated by lagging
- `cbind(y.lost, y.lagged, y.future)` is the complete series after creation and addition of the lagged variables.

**Examples**

```r
set.seed(4711)
# generate bivariate time series
y<-round(matrix(10*rnorm(36), nrow=3, ncol=2))
y
# define new lagged variables (y3 and y4) with seasonalities 6 and 12
lagging <- c(1, 3, (6-1), 2, 4, (12-1))
season.lagging(y, lagging)
```

**Description**

Function to condensate (and/or) print out matrix polynomial leaving out empty lag matrices and, if specified, the leading (unity) matrix.

**Usage**

```r
short.form(poly = NULL, name = "Lag=", leading = TRUE, tail = FALSE, digits = 6)
```
Arguments

poly matrix polynomium (0-1 array as constructed by define.model, for example, or array of reals as estimated by marima).

name character string used as header in output (default='lag').

leading TRUE/FALSE. If leading=FALSE the leading (unity matrix) is to be left out/suppressed.

tail TRUE/FALSE. If TRUE and the ar/ma-model only consists of coefficient matrices where all coefficients (except the leading unity matrix) are all zero a first order coefficient matrix (being zero) is retained (in order to avoid a model containing only the leading unity matrix).

If tail=TRUE and the coefficients in the first coefficient matrix (after the leading unity matrix) are all zero, the leading unity matrix is always retained.

digits the number of digits retained by short.form (default=6).

Examples

Model<-define.model(kvar=4, ar=c(1, 2, 4), ma=c(1), reg.var=4)
short.form(Model$ar.pattern)
short.form(Model$ma.pattern)
short.form(Model$ar.pattern, leading=FALSE)
short.form(Model$ar.pattern, leading=FALSE)

#
M<-define.model(kvar=4, ma=c(1))
short.form(M$ar.pattern)
short.form(M$ar.pattern, tail=TRUE)
short.form(M$ar.pattern, leading=FALSE, tail=TRUE)
Index

arma.filter, 2
arma.forecast, 3
austr, 5
C20, 6
check.one, 8
define.dif, 9
define.model, 10
define.sum, 13
forec.var, 13
inverse.form, 14
lead.one, 15
marima, 15
marima.sim, 19
pol.inv, 20
pol.mul, 21
pol.order, 22
print.marima, 23
rand.shock, 23
Results, 24
season.lagging, 24
short.form, 25