Package ‘timsac’

March 30, 2023

Version 1.3.8
Title Time Series Analysis and Control Package
Author The Institute of Statistical Mathematics
Maintainer Masami Saga <msaga@mtb.biglobe.ne.jp>
Depends R (>= 3.0.0)
Imports graphics, grDevices, stats
License GPL (>= 2)
MailingList Please send bug reports to ismrp@jasp.ism.ac.jp.
NeedsCompilation yes

R topics documented:

timsac-package .................................................. 2
Airpollution ....................................................... 3
Amerikamaru ...................................................... 3
armafit ............................................................ 4
auspec ............................................................ 5
autocor ............................................................ 6
autoarmafit ......................................................... 7
baysea ............................................................. 8
bispec ............................................................ 10
bispecData ........................................................ 11
blocar ............................................................ 11
blomar ............................................................ 13
Bilsallfood ......................................................... 14
bsubst ............................................................ 15
Canadianlynx ...................................................... 17
canarm ............................................................ 17
canoca ............................................................ 19
covgen ............................................................ 20
decomp ............................................................ 21
timsac-package

Description

R functions for statistical analysis and control of time series.
Details

This package provides functions for statistical analysis, prediction and control of time series. The original TIMSAC (TIME Series Analysis and Control) or TIMSAC-72 was published in Akaike and Nakagawa (1972). After that, TIMSAC-74, TIMSAC-78 and TIMSAC-84 were published as the TIMSAC series in Computer Science Monograph.

For overview of models and information criteria for model selection, see ../doc/timsac-guide_e.pdf or ../doc/timsac-guide_j.pdf (in Japanese).

References


<table>
<thead>
<tr>
<th>Airpollution</th>
<th>Airpollution Data</th>
</tr>
</thead>
</table>

Description

An airpollution data for testing perars.

Usage

data(Airpollution)

Format

A time series of 372 observations.

Source

Amerikamaru  

**Amerikamaru Data**  

**Description**  
A multivariate non-stationary data for testing `blomar`.  

**Usage**  
```r  
data(Amerikamaru)  
```

**Format**  
A 2-dimensional array with 896 observations on 2 variables.  

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[. 1]</td>
<td>rudder</td>
</tr>
<tr>
<td>[. 2]</td>
<td>yawing</td>
</tr>
</tbody>
</table>

**Source**  

---

armafit  

**ARMA Model Fitting**  

**Description**  
Fit an ARMA model with specified order by using DAVIDON’s algorithm.  

**Usage**  
```r  
armafit(y, model.order)  
```

**Arguments**  
- `y`  
a univariate time series.  
- `model.order`  
a numerical vector of the form `c(ar, ma)` which gives the order to be fitted successively.  

**Details**  
The maximum likelihood estimates of the coefficients of a scalar ARMA model  

\[ y(t) - a(1)y(t-1) - \ldots - a(p)y(t-p) = u(t) - b(1)u(t-1) - \ldots - b(q)u(t-q) \]

of a time series \( y(t) \) are obtained by using DAVIDON’s algorithm. Pure autoregression is not allowed.
Value

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arcoef</td>
<td>maximum likelihood estimates of AR coefficients.</td>
</tr>
<tr>
<td>macoef</td>
<td>maximum likelihood estimates of MA coefficients.</td>
</tr>
<tr>
<td>arstd</td>
<td>standard deviation (AR).</td>
</tr>
<tr>
<td>mastd</td>
<td>standard deviation (MA).</td>
</tr>
<tr>
<td>v</td>
<td>innovation variance.</td>
</tr>
<tr>
<td>aic</td>
<td>AIC.</td>
</tr>
<tr>
<td>grad</td>
<td>final gradient.</td>
</tr>
</tbody>
</table>

References


Examples

```r
# "arima.sim" is a function in "stats".
# Note that the sign of MA coefficient is opposite from that in "timsac".
y <- arima.sim(list(order=c(2,0,1), ar=c(0.64,-0.8), ma=-0.5), n = 1000)
z <- armafit(y, model.order = c(2,1))
z$arcoef
z$macoef
```

---

**auspec**

*Power Spectrum*

Description

Compute power spectrum estimates for two trigonometric windows of Blackman-Tukey type by Goertzel method.

Usage

```r
auspec(y, lag = NULL, window = "Akaike", log = FALSE, plot = TRUE)
```

Arguments

- `y`: a univariate time series.
- `lag`: maximum lag. Default is $2\sqrt{n}$, where $n$ is the length of time series $y$.
- `window`: character string giving the definition of smoothing window. Allowed strings are "Akaike" (default) or "Hanning".
- `log`: logical. If TRUE, the spectrum `spec` is plotted as log(`spec`).
- `plot`: logical. If TRUE (default), the spectrum `spec` is plotted.

Details

---
Value

spec  spectrum smoothing by 'window'
stat  test statistics.

References


Examples

```r
y <- arima.sim(list(order=c(2,0,0), ar=c(0.64,-0.8)), n = 200)
auspec(y, log = TRUE)
```

Description

Estimate autocovariances and autocorrelations.

Usage

```r
autcor(y, lag = NULL, plot = TRUE, lag_axis = TRUE)
```

Arguments

- `y`  a univariate time series.
- `lag`  maximum lag. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.
- `plot`  logical. If TRUE (default), autocorrelations are plotted.
- `lag_axis`  logical. If TRUE (default) with plot = TRUE, x-axis is drawn.

Value

- `acov`  autocovariances.
- `acor`  autocorrelations (normalized covariances).
- `mean`  mean of $y$.
autoarmafit

References


Examples

```r
# Example 1 for the normal distribution
y <- rnorm(200)
autcor(y, lag_axis = FALSE)

# Example 2 for the ARIMA model
y <- arima.sim(list(order=c(2,0,0), ar=c(0.64,-0.8)), n = 200)
autcor(y, lag = 20)
```

autoarmafit  

*Automatic ARMA Model Fitting*

Description

Provide an automatic ARMA model fitting procedure. Models with various orders are fitted and the best choice is determined with the aid of the statistics AIC.

Usage

```r
autoarmafit(y, max.order = NULL)
```

Arguments

- `y`: a univariate time series.
- `max.order`: upper limit of AR order and MA order. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.

Details

The maximum likelihood estimates of the coefficients of a scalar ARMA model

$$y(t) - a(1)y(t-1) - \ldots - a(p)y(t-p) = u(t) - b(1)u(t-1) - \ldots - b(q)u(t-q)$$

of a time series $y(t)$ are obtained by using DAVIDON’s variance algorithm. Where $p$ is AR order, $q$ is MA order and $u(t)$ is a zero mean white noise. Pure autoregression is not allowed.

Value

- `model`: a list with components `arcoef` (Maximum likelihood estimates of AR coefficients), `macoef` (Maximum likelihood estimates of MA coefficients), `arstd` (AR standard deviation), `mastd` (MA standard deviation), `v` (Innovation variance), `aic` ($\text{AIC} = n \log(\det(v)) + 2(p + q)$) and `grad` (Final gradient) in AIC increasing order.
baysea

Bayesian Seasonal Adjustment Procedure

Description
Decompose a nonstationary time series into several possible components.

Usage
baysea(y, period = 12, span = 4, shift = 1, forecast = 0, trend.order = 2,
seasonal.order = 1, year = 0, month = 1, out = 0, rigid = 1,
zersum = 1, delta = 7, alpha = 0.01, beta = 0.01, gamma = 0.1,
spec = TRUE, plot = TRUE, separate.graphics = FALSE)

Arguments
y a univariate time series.
period number of seasonals within a period.
span number of periods to be processed at one time.
shift number of periods to be shifted to define the new span of data.
forecast length of forecast at the end of data.
trend.order order of differencing of trend.
seasonal.order order of differencing of seasonal. seasonal.order is smaller than or equal to span.
year trading-day adjustment option.
  = 0 : without trading day adjustment
  > 0 : with trading day adjustment
    (the series is supposed to start at this year)
month number of the month in which the series starts. If year=0 this parameter is ignored.
out outlier correction option.
0: without outlier detection
1: with outlier detection by marginal probability
2: with outlier detection by model selection

rigid controls the rigidity of the seasonal component. More rigid seasonal with larger than rigid.
zersum controls the sum of the seasonals within a period.
delta controls the leap year effect.
alpha controls prior variance of initial trend.
beta controls prior variance of initial seasonal.
gamma controls prior variance of initial sum of seasonal.
spec logical. If TRUE (default), estimate spectra of irregular and differenced adjusted.
plot logical. If TRUE (default), plot trend, adjust, smoothed, season and irregular.
separate.graphics logical. If TRUE, a graphic device is opened for each graphics display.

Details

This function realized a decomposition of time series y into the form

\[ y(t) = T(t) + S(t) + I(t) + TDC(t) + OCF(t) \]

where T(t) is trend component, S(t) is seasonal component, I(t) is irregular, TDC(t) is trading day factor and OCF(t) is outlier correction factor. For the purpose of comparison of models the criterion ABIC is defined

\[ ABIC = -2 \log(\text{maximum likelihood of the model}). \]

Smaller value of ABIC represents better fit.

Value

outlier outlier correction factor.
trend trend.
season seasonal.
tday trading day component if year > 0.
irregular = y - trend - season - tday - outlier.
adj = trend - irregular.
smoothed = trend + season + tday.
avgABIC averaged ABIC.
irregular.spec a list with components acov (autocovariances), acor (normalized covariances), mean, v (innovation variance), aic (AIC), parcor (partial autocorrelation) and rspec (rational spectrum) of irregular if spec = TRUE.
adjusted.spec a list with components acov, acor, mean, v, aic, parcor and rspec of differenced adjusted series if spec = TRUE.
differenced.trend
   a list with components acov, acor, mean, v, aic and parcor of differenced trend series if spec = TRUE.
differenced.season
   a list with components acov, acor, mean, v, aic and parcor of differenced seasonal series if spec = TRUE.

References

Examples
```r
data(LaborData)
baysea(LaborData, forecast = 12)
```

---

**bispec**

**Bispectrum**

**Description**
Compute bi-spectrum using the direct Fourier transform of sample third order moments.

**Usage**
```r
bispec(y, lag = NULL, window = "Akaike", log = FALSE, plot = TRUE)
```

**Arguments**
- `y` : a univariate time series.
- `lag` : maximum lag. Default is $2 \sqrt{n}$, where $n$ is the length of the time series $y$.
- `window` : character string giving the definition of smoothing window. Allowed strings are "Akaike" (default) or "Hanning".
- `log` : logical. If TRUE, the spectrum $pspec$ is plotted as $\log(pspec)$.
- `plot` : logical. If TRUE (default), the spectrum $pspec$ is plotted.

**Details**
- Hanning Window: $a_1(0)=0.5$, $a_1(1)=a_1(-1)=0.25$, $a_1(2)=a_1(-2)=0$
- Akaike Window: $a_2(0)=0.625$, $a_2(1)=a_2(-1)=0.25$, $a_2(2)=a_2(-2)=-0.0625$
bispecData

Value
   pspec      power spectrum smoothed by 'window'.
   sig        significance.
   cohe       coherence.
   breal      real part of bispectrum.
   bimag      imaginary part of bispectrum.
   exval      approximate expected value of coherence under Gaussian assumption.

References


Examples

data(bispecData)
bispec(bispecData, lag = 30)

bispecData                   Univariate Test Data

Description

A univariate data for testing bispec and thirmo.

Usage

data(bispecData)

Format

A time series of 1500 observations.

Source

blocar

Bayesian Method of Locally Stationary AR Model Fitting: Scalar Case

Description

Locally fit autoregressive models to non-stationary time series by a Bayesian procedure.

Usage

blocar(y, max.order = NULL, span, plot = TRUE)

Arguments

y          a univariate time series.
max.order  upper limit of the order of AR model. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.
span       length of basic local span.
pplot      logical. If TRUE (default), spectrums pspec are plotted.

Details

The basic AR model of scalar time series $y(t)(t = 1, \ldots, n)$ is given by

$$y(t) = a(1)y(t-1) + a(2)y(t-2) + \ldots + a(p)y(t-p) + u(t),$$

where $p$ is order of the model and $u(t)$ is Gaussian white noise with mean $0$ and variance $v$. At each stage of modeling of locally AR model, a two-step Bayesian procedure is applied

1. Averaging of the models with different orders fitted to the newly obtained data.
2. Averaging of the models fitted to the present and preceding spans.

AIC of the model fitted to the new span is defined by

$$AIC = ns \log(sd) + 2k,$$

where $ns$ is the length of new data, $sd$ is innovation variance and $k$ is the equivalent number of parameters, defined as the sum of squares of the Bayesian weights. AIC of the model fitted to the preceding spans are defined by

$$AIC(j + 1) = ns \log(sd(j)) + 2,$$

where $sd(j)$ is the prediction error variance by the model fitted to $j$ periods former span.

Value

var        variance.
aic        AIC.
bweight  Bayesian weight.
pacoeft  partial autocorrelation.
arcoeft  coefficients ( average by the Bayesian weights ).
v       innovation variance.
init    initial point of the data fitted to the current model.
end     end point of the data fitted to the current model.
pspec   power spectrum.

References


Examples

data(locarData)
z <- blocar(locarData, max.order = 10, span = 300)
z$arcoef

blomar  Bayesian Method of Locally Stationary Multivariate AR Model Fitting

Description

Locally fit multivariate autoregressive models to non-stationary time series by a Bayesian procedure.

Usage

blomar(y, max.order = NULL, span)

Arguments

y       A multivariate time series.
max.order  upper limit of the order of AR model, less than or equal to \( n/2d \) where \( n \) is the length and \( d \) is the dimension of the time series \( y \). Default is \( \min(2\sqrt{n}, n/2d) \).
span     length of basic local span. Let \( m \) denote max.order, if \( n - m - 1 \) is less than or equal to span or \( n - m - 1 - \text{span} \) is less than \( 2md \), span is \( n - m \).
Details

The basic AR model is given by

\[ y(t) = A(1)y(t-1) + A(2)y(t-2) + \ldots + A(p)y(t-p) + u(t), \]

where \( p \) is order of the AR model and \( u(t) \) is innovation variance \( \nu \).

Value

- **mean**: mean.
- **var**: variance.
- **bweight**: Bayesian weight.
- **aic**: AIC with respect to the present data.
- **arcoef**: AR coefficients. \( \text{arcoef}[[m]][i,j,k] \) shows the value of \( i \)-th row, \( j \)-th column, \( k \)-th order of \( m \)-th model.
- **v**: innovation variance.
- **eaic**: equivalent AIC of Bayesian model.
- **init**: start point of the data fitted to the current model.
- **end**: end point of the data fitted to the current model.

References


Examples

```r
data(Amerikamaru)
blomar(Amerikamaru, max.order = 10, span = 300)
```

---

**Blsallfood**  

**Blsallfood Data**

Description

The BLSALLFOOD data. (the Bureau of Labor Statistics, all employees in food industries, January 1967 - December 1979)

Usage

```r
data(Blsallfood)
```
Format

A time series of 156 observations.

Source


\begin{center}
\textbf{bsubst} \hspace{1cm} \textit{Bayesian Type All Subset Analysis}
\end{center}

Description

Produce Bayesian estimates of time series models such as pure AR models, AR models with non-linear terms, AR models with polynomial type mean value functions, etc. The goodness of fit of a model is checked by the analysis of several steps ahead prediction errors.

Usage

\begin{verbatim}
bsubst(y, mtype, lag = NULL, nreg, reg = NULL, term.lag = NULL, cstep = 5,
      plot = TRUE)
\end{verbatim}

Arguments

- \textbf{y}: a univariate time series.
- \textbf{mtype}: model type. Allowed values are
  \begin{enumerate}
    \item 1: autoregressive model,
    \item 2: polynomial type non-linear model (lag’s read in),
    \item 3: polynomial type non-linear model (lag’s automatically set),
    \item 4: AR-model with polynomial mean value function,
    \item 5,6,7: originally defined but omitted here.
  \end{enumerate}
- \textbf{lag}: maximum time lag. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.
- \textbf{nreg}: number of regressors.
- \textbf{reg}: specification of regressor (mtype = 2).
  \begin{verbatim}
i-th regressor is defined by \( z(n - L1(i)) \times z(n - L2(i)) \times z(n - L3(i)) \), where
\end{verbatim}
  \begin{itemize}
    \item \( L1(i) \) is \( \text{reg}(1,i) \),
    \item \( L2(i) \) is \( \text{reg}(2,i) \) and
    \item \( L3(i) \) is \( \text{reg}(3,i) \).
  \end{itemize}
  0-lag term \( z(n - 0) \) is replaced by the constant 1.
- \textbf{term.lag}: maximum time lag specify the regressors \( (L1(i), L2(i), L3(i)) \) (i=1,...,nreg)
  (mtype = 3).
  \begin{verbatim}
  term.lag[1]: maximum time lag of linear term
  term.lag[2]: maximum time lag of squared term
  term.lag[3]: maximum time lag of quadratic crosses term
\end{verbatim}
term.lag[4]: maximum time lag of cubic term
term.lag[5]: maximum time lag of cubic cross term.
cstep: prediction errors checking (up to cstep-steps ahead) is requested. (mtype = 1, 2, 3).
plot: logical. If TRUE (default), daic, perr and peautcor are plotted.

Details

The AR model is given by (mtype = 2)

\[ y(t) = a(1)y(t-1) + \ldots + a(p)y(t-p) + u(t). \]

The non-linear model is given by (mtype = 2, 3)

\[ y(t) = a(1)z(t,1) + a(2)z(t,2) + \ldots + a(p)z(t,p) + u(t). \]

Where \( p \) is AR order and \( u(t) \) is Gaussian white noise with mean 0 and variance \( v(p) \).

Value

- ymean: mean of \( y \).
- yvar: variance of \( y \).
- v: innovation variance.
- aic: AIC(m), (m=0, \ldots nreg).
- aicmin: minimum AIC.
- daic: AIC(m)-aicmin (m=0, \ldots nreg).
- order.maice: order of minimum AIC.
- v.maice: innovation variance attained at order.maice.
- arcoef.maice: AR coefficients attained at order.maice.
- v.bay: residual variance of Bayesian model.
- aic.bay: AIC of Bayesian model.
- np.bay: equivalent number of parameters.
- arcoef.bay: AR coefficients of Bayesian model.
- ind.c: index of parcor2 in order of increasing magnitude.
- parcor2: square of partial correlations (normalized by multiplying N).
- damp: binomial type damper.
- bweight: final Bayesian weights of partial correlations.
- parcor.bay: partial correlations of the Bayesian model.
- eicmin: minimum EIC.
- esum: whole subset regression models.
- npmean: mean of number of parameter.
- npmean.nreg = npmean / nreg.
Canadian lynx data

Description
A time series of Canadian lynx data for testing unimar, unibar, bsubst and exsar.

Usage
data(Canadianlynx)

Format
A time series of 114 observations.

Source
Description

Fit an ARMA model to stationary scalar time series through the analysis of canonical correlations between the future and past sets of observations.

Usage

canarm(y, lag = NULL, max.order = NULL, plot = TRUE)

Arguments

y a univariate time series.
lag maximum lag. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.
max.order upper limit of AR order and MA order, must be less than or equal to lag. Default is lag.
plot logical. If TRUE (default), parcor is plotted.

Details

The ARMA model of stationary scalar time series $y(t)(t = 1, \ldots, n)$ is given by

$$y(t) - a(1)y(t - 1) - \ldots - a(p)y(t - p) = u(t) - b(1)u(t - 1) - \ldots - b(q)u(t - q),$$

where $p$ is AR order and $q$ is MA order.

Value

arinit AR coefficients of initial AR model fitting by the minimum AIC procedure.
v innovation vector.
aic AIC.
aicmin minimum AIC.
order.maice order of minimum AIC.
parcor partial autocorrelation.
nc total number of case.
future number of present and future variables.
past number of present and past variables.
cweight future set canonical weight.
canocoeef canonical R.
canocoeef2 R-squared.
chisquar chi-square.
canoca

Description

Analyze canonical correlation of a d-dimensional multivariate time series.

Usage

canoca(y)

Arguments

y        a multivariate time series.

Details

First AR model is fitted by the minimum AIC procedure. The results are used to ortho-normalize the present and past variables. The present and future variables are tested successively to decide on the dependence of their predictors. When the last DIC (=chi-square - 2.0*N.D.F.) is negative the predictor of the variable is decided to be linearly dependent on the antecedents.
Value

- aic: AIC.
- aicmin: minimum AIC.
- order.maice: MAICE AR model order.
- v: innovation variance.
- arcoef: autoregressive coefficients. arcoef[i,j,k] shows the value of i-th row, j-th column, k-th order.
- nc: number of cases.
- future: number of variable in the future set.
- past: number of variables in the past set.
- cweight: future set canonical weight.
- canocoef: canonical R.
- canocoef2: R-squared.
- chisquar: chi-square.
- ndf: N.D.F.
- dic: DIC.
- dicmin: minimum DIC.
- order.dicmin: order of minimum DIC.
- matF: the transition matrix F.
- vectH: structural characteristic vector H of the canonical Markovian representation.
- matG: the estimate of the input matrix G.
- vectF: matrix F in vector form.

References


Examples

```r
ar <- array(0, dim = c(3,3,2))
ar[, , 1] <- matrix(c(0.4, 0, 0.3,
                     0.2, -0.1, -0.5,
                     0.3, 0.1, 0), nrow = 3, ncol = 3, byrow= TRUE)
ar[, , 2] <- matrix(c(0, -0.3, 0.5,
                     0.7, -0.4, 1,
                     0, -0.5, 0.3), nrow = 3, ncol = 3, byrow = TRUE)
x <- matrix(rnorm(1000*3), nrow = 1000, ncol = 3)
y <- mfilter(x, ar, "recursive")
z <- canoca(y)
z$arcoef
```
covgen

Covariance Generation

Description

Produce the Fourier transform of a power gain function in the form of an autocovariance sequence.

Usage

covgen(lag, f, gain, plot = TRUE)

Arguments

lag
desired maximum lag of covariance.

f
frequency \( f[i] \) \( (i = 1, \ldots, k) \), where \( k \) is the number of data points. By definition \( f[1] = 0.0 \) and \( f[k] = 0.5 \), \( f[i]'s \) are arranged in increasing order.

gain
power gain of the filter at the frequency \( f[i] \).

plot
logical. If TRUE (default), autocorrelations are plotted.

Value

acov
autocovariance.

acor
autocovariance normalized.

References


Examples

```r
spec <- raspec(h = 100, var = 1, arcoef = c(0.64,-0.8), plot = FALSE)
covgen(lag = 100, f = 0:100/200, gain = spec)
```

decomp

Time Series Decomposition (Seasonal Adjustment) by Square-Root Filter

Description

Decompose a nonstationary time series into several possible components by square-root filter.
Usage

decomp(y, trend.order = 2, ar.order = 2, seasonal.order = 1,
      period = 1, log = FALSE, trade = FALSE, diff = 1,
      miss = 0, omax = 99999.9, plot = TRUE)

Arguments

y
  a univariate time series with or without the tsp attribute.
trend.order
  trend order (1, 2 or 3).
ar.order
  AR order (less than 11, try 2 first).
seasonal.order
  seasonal order (0, 1 or 2).
period
  number of seasons in one period. If the tsp attribute of y is not NULL, frequency(y).
log
  logical; if TRUE, a log scale is in use.
trade
  logical; if TRUE, the model including trading day effect component is considered, where tsp(y) is not null and frequency(y) is 4 or 12.
diff
  numerical differencing (1 sided or 2 sided).
miss
  missing data flag.
    = 0 : no consideration
    > 0 : values which are greater than omax are treated as missing data
    < 0 : values which are less than omax are treated as missing data
omax
  maximum or minimum data value (if miss > 0 or miss < 0).
plot
  logical. If TRUE (default), trend, seasonal, ar and trad are plotted.

Details

The Basic Model

\[ y(t) = T(t) + AR(t) + S(t) + TD(t) + W(t) \]

where \( T(t) \) is trend component, \( AR(t) \) is AR process, \( S(t) \) is seasonal component, \( TD(t) \) is trading day factor and \( W(t) \) is observational noise.

Component Models

- Trend component (trend.order m1)
  \[ m1 = 1 : T(t) = T(t - 1) + v1(t) \]
  \[ m1 = 2 : T(t) = 2T(t - 1) - T(t - 2) + v1(t) \]
  \[ m1 = 3 : T(t) = 3T(t - 1) - 3T(t - 2) + T(t - 2) + v1(t) \]

- AR component (ar.order m2)
  \[ AR(t) = a(1)AR(t - 1) + \ldots + a(m2)AR(t - m2) + v2(t) \]
• Seasonal component (seasonal.order k, frequency f)
  \[ k = 1 : S(t) = -S(t - 1) - \ldots - S(t - f + 1) + v3(t) \]
  \[ k = 2 : S(t) = -2S(t - 1) - \ldots - f S(t - f + 1) - \ldots - S(t - 2f + 2) + v3(t) \]

• Trading day effect
  \[ T\!D(t) = b(1)TRADE(t, 1) + \ldots + b(7)TRADE(t, 7) \]
  where \( TRADE(t, i) \) is the number of \( i \)-th days of the week in \( t \)-th data and \( b(1) + \ldots + b(7) = 0 \).

Value
- trend: trend component.
- seasonal: seasonal component.
- ar: AR process.
- trad: trading day factor.
- noise: observational noise.
- aic: AIC.
- lkhd: likelihood.
- sigma2: \( \sigma^2 \).
- tau1: system noise variances \( \nu_1 \).
- tau2: system noise variances \( \nu_2 \) or \( \nu_3 \).
- tau3: system noise variances \( \nu_3 \).
- arcoef: vector of AR coefficients.
- tdf: trading day factor. \( tdf(i) \) (i=1,7) are from Sunday to Saturday sequentially.

References


Examples

```r
data(Blsallfood)
y <- ts(Blsallfood, start=c(1967,1), frequency=12)
z <- decomp(y, trade = TRUE)
z$aic
z$lkhd
z$sigma2
z$tau1
z$tau2
z$tau3
```
Description

Produce exact maximum likelihood estimates of the parameters of a scalar AR model.

Usage

```r
exsar(y, max.order = NULL, plot = FALSE)
```

Arguments

- `y`: a univariate time series.
- `max.order`: upper limit of AR order. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.
- `plot`: logical. If `TRUE`, $\text{daic}$ is plotted.

Details

The AR model is given by

$$y(t) = a(1)y(t-1) + \ldots + a(p)y(t-p) + u(t)$$

where $p$ is AR order and $u(t)$ is a zero mean white noise.

Value

- `mean`: mean.
- `var`: variance.
- `v`: innovation variance.
- `aic`: AIC.
- `aicmin`: minimum AIC.
- `daic`: AIC-aicmin.
- `order.maice`: order of minimum AIC.
- `v.maice`: MAICE innovation variance.
- `arcoefficient.maice`: MAICE AR coefficients.
- `v.mle`: maximum likelihood estimates of innovation variance.
- `arcoefficient.mle`: maximum likelihood estimates of AR coefficients.

References

fftcor

Examples

data(Canadianlynx)
z <- exsar(Canadianlynx, max.order = 14)
z$arcoef.maice
z$arcoef.mle

fftcor Auto And/Or Cross Correlations via FFT

Description

Compute auto and/or cross covariances and correlations via FFT.

Usage

fftcor(y, lag = NULL, isw = 4, plot = TRUE, lag_axis = TRUE)

Arguments

y data of channel X and Y (data of channel Y is given for isw = 2 or 4 only).
lag maximum lag. Default is $2\sqrt{n}$, where $n$ is the length of the time series y.
isw numerical flag giving the type of computation.
  1: auto-correlation of X (one-channel)
  2: auto-correlations of X and Y (two-channel)
  4: auto- and cross-correlations of X and Y (two-channel)
plot logical. If TRUE (default), cross-correlations are plotted.
lag_axis logical. If TRUE (default) with plot=TRUE, x-axis is drawn.

Value

acov auto-covariance.
ccov12 cross-covariance.
ccov21 cross-covariance.
acor auto-correlation.
ccor12 cross-correlation.
ccor21 cross-correlation.
mean mean.

References

Examples

# Example 1
x <- rnorm(200)
y <- rnorm(200)
xy <- array(c(x,y), dim = c(200,2))
fftcor(xy, lag_axis = FALSE)

# Example 2
xorg <- rnorm(1003)
x <- matrix(0, nrow = 1000, ncol = 2)
x[, 1] <- xorg[1:1000]
x[, 2] <- xorg[4:1003] + 0.5*rt(1000)
fftcor(x, lag = 20)

fpeaut

FPE Auto

Description

Perform FPE(Final Prediction Error) computation for one-dimensional AR model.

Usage

fpeaut(y, max.order = NULL)

Arguments

y a univariate time series.
max.order upper limit of model order. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.

Details

The AR model is given by

$$y(t) = a(1)y(t-1) + \ldots + a(p)y(t-p) + u(t)$$

where $p$ is AR order and $u(t)$ is a zero mean white noise.

Value

ordermin order of minimum FPE.
best.ar AR coefficients with minimum FPE.
sigma2m = sigma2(ordermin).
fpemin minimum FPE.
rfpemin minimum RFPE.
ofpe OFPE.
arcoef AR coefficients.
sigma2 $\sigma^2$.
fpe FPE (Final Prediction Error).
rfpe RFPE.
parcor partial correlation.
chi2 chi-squared.

References

Examples

```r
y <- arima.sim(list(order=c(2,0,0), ar=c(0.64,-0.8)), n = 200)
fpec(y, max.order = 20)
```

---

**Description**

Perform AR model fitting for control.

**Usage**

```r
fpec(y, max.order = NULL, control = NULL, manip = NULL)
```

**Arguments**

- `y`: a multivariate time series.
- `max.order`: upper limit of model order. Default is $2\sqrt{n}$, where $n$ is the length of time series $y$.
- `control`: controlled variables. Default is $c(1 : d)$, where $d$ is the dimension of the time series $y$.
- `manip`: manipulated variables. Default number of manipulated variable is 0.

**Value**

- `cov`: covariance matrix rearrangement.
- `fpec`: FPEC (AR model fitting for control).
- `rfpec`: RFPEC.
- `aic`: AIC.
- `ordermin`: order of minimum FPEC.
fpecmin          minimum FPEC.
rfpecmin          minimum RFPEC.
aicmin          minimum AIC.
perr              prediction error covariance matrix.
arcoef          a set of coefficient matrices. arcoef[i,j,k] shows the value of i-th row, j-th column, k-th order.

References

Examples
ar <- array(0, dim = c(3,3,2))
ar[, , 1] <- matrix(c(0.4, 0, 0.3, 0.2, -0.1, -0.5, 0.3, 0.1, 0), nrow = 3, ncol = 3, byrow = TRUE)
ar[, , 2] <- matrix(c(0, -0.3, 0.5, 0.7, -0.4, 1, 0, -0.5, 0.3), nrow = 3, ncol = 3, byrow = TRUE)
x <- matrix(rnorm(200*3), nrow = 200, ncol = 3)
y <- mfilter(x, ar, "recursive")
fpec(y, max.order = 10)

LaborData          Labor force Data

Description
Labor force U.S. unemployed 16 years or over (1972-1978) data.

Usage
data(LaborData)

Format
A time series of 72 observations.

Source
**locarData**

**Non-stationary Test Data**

**Description**

A non-stationary data for testing `mlocar` and `blocar`.

**Usage**

```r
data(locarData)
```

**Format**

A time series of 1000 observations.

**Source**


---

**markov**

**Maximum Likelihood Computation of Markovian Model**

**Description**

Compute maximum likelihood estimates of Markovian model.

**Usage**

```r
markov(y)
```

**Arguments**

- `y`: a multivariate time series.

**Details**

This function is usually used with `simcon`. 
**Value**

id  \( id[i] = 1 \) means that the \( i \)-th row of \( F \) contains free parameters.

ir  \( ir[i] \) denotes the position of the last non-zero element within the \( i \)-th row of \( F \).

ij  \( ij[i] \) denotes the position of the \( i \)-th non-trivial row within \( F \).

ik  \( ik[i] \) denotes the number of free parameters within the \( i \)-th non-trivial row of \( F \).

grad  gradient vector.

matFi  initial estimate of the transition matrix \( F \).

matF  transition matrix \( F \).

matG  input matrix \( G \).

davvar  DAVIDON variance.

arcoef  AR coefficient matrices. \( arcoef[i,j,k] \) shows the value of \( i \)-th row, \( j \)-th column, \( k \)-th order.

impulse  impulse response matrices.

macoef  MA coefficient matrices. \( macoef[i,j,k] \) shows the value of \( i \)-th row, \( j \)-th column, \( k \)-th order.

v  innovation variance.

aic  AIC.

**References**


**Examples**

```r
x <- matrix(rnorm(1000*2), nrow = 1000, ncol = 2)
ma <- array(0, dim = c(2,2,2))
ma[, , 1] <- matrix(c( -1.0, 0.0,
                      0.0, -1.0), nrow = 2, ncol = 2, byrow = TRUE)
ma[, , 2] <- matrix(c( -0.2, 0.0,
                      -0.1, -0.3), nrow = 2, ncol = 2, byrow = TRUE)
y <- mfilter(x, ma, "convolution")
ar <- array(0, dim = c(2,2,3))
ar[, , 1] <- matrix(c( -1.0, 0.0,
                     0.0, -1.0), nrow = 2, ncol = 2, byrow = TRUE)
ar[, , 2] <- matrix(c( -0.5, -0.2,
                     -0.2, -0.5), nrow = 2, ncol = 2, byrow = TRUE)
ar[, , 3] <- matrix(c( -0.3, -0.05,
                     -0.1, -0.30), nrow = 2, ncol = 2, byrow = TRUE)
z <- mfilter(y, ar, "recursive")
markov(z)
```
**Linear Filtering on a Multivariate Time Series**

**Description**

Applies linear filtering to a multivariate time series.

**Usage**

```r
mfilter(x, filter, method = c("convolution","recursive"), init)
```

**Arguments**

- `x`: a multivariate \((m\times n)\) time series \(x[n,m]\).
- `filter`: an array of filter coefficients. \(filter[i,j,k]\) shows the value of \(i\)-th row, \(j\)-th column, \(k\)-th order.
- `method`: either "convolution" or "recursive" (and can be abbreviated). If "convolution" a moving average is used; if "recursive" an autoregression is used. For convolution filters, the filter coefficients are for past value only.
- `init`: specifies the initial values of the time series just prior to the start value, in reverse time order. The default is a set of zeros.

**Details**

This is a multivariate version of "filter" function. Missing values are allowed in 'x' but not in 'filter' (where they would lead to missing values everywhere in the output). Note that there is an implied coefficient 1 at lag 0 in the recursive filter, which gives

\[
y[i,]' = x[i,]' + f[i,1] \times y[i-1,]' + \ldots + f[i,p] \times y[i-p,]'
\]

No check is made to see if recursive filter is invertible: the output may diverge if it is not. The convolution filter is

\[
y[i,]' = f[i,1] \times x[i,]' + \ldots + f[i,p] \times x[i-p+1,]'
\]

**Value**

`mfilter` returns a time series object.

**Note**

'convolve(, type="filter")' uses the FFT for computations and so may be faster for long filters on univariate time series (and so the time alignment is unclear), nor does it handle missing values. 'filter' is faster for a filter of length 100 on a series 1000, for examples.

**See Also**

`convolve, arima.sim`
Examples

#AR model simulation
ar <- array(0, dim = c(3,3,2))
ar[, , 1] <- matrix(c(0.4, 0, 0.3,
                     0.2, -0.1, -0.5,
                     0.3, 0.1, 0), nrow = 3, ncol = 3, byrow = TRUE)
ar[, , 2] <- matrix(c(0, -0.3, 0.5,
                     0.7, -0.4, 1,
                     0, -0.5, 0.3), nrow = 3, ncol = 3, byrow = TRUE)
x <- matrix(rnorm(100*3), nrow = 100, ncol = 3)
y <- mfilter(x, ar, "recursive")

#Back to white noise
ma <- array(0, dim = c(3,3,3))
ma[, , 1] <- diag(3)
ma[, , 2] <- -ar[, , 1]
ma[, , 3] <- -ar[, , 2]
z <- mfilter(y, ma, "convolution")
mulcor(z)

#AR-MA model simulation
x <- matrix(rnorm(1000*2), nrow = 1000, ncol = 2)
ma <- array(0, dim = c(2,2,2))
ma[, , 1] <- matrix(c(-1.0, 0.0,
                     0.0, -1.0), nrow = 2, ncol = 2, byrow = TRUE)
ma[, , 2] <- matrix(c(-0.2, 0.0,
                     -0.1, -0.3), nrow = 2, ncol = 2, byrow = TRUE)
y <- mfilter(x, ma, "convolution")
ar <- array(0, dim = c(2,2,3))
ar[, , 1] <- matrix(c(-1.0, 0.0,
                     0.0, -1.0), nrow = 2, ncol = 2, byrow = TRUE)
ar[, , 2] <- matrix(c(-0.5, -0.2,
                     -0.2, -0.5), nrow = 2, ncol = 2, byrow = TRUE)
ar[, , 3] <- matrix(c(-0.3, -0.05,
                     -0.1, -0.30), nrow = 2, ncol = 2, byrow = TRUE)
z <- mfilter(y, ar, "recursive")

mlocar

Minimum AIC Method of Locally Stationary AR Model Fitting: Scalar Case

Description
Locally fit autoregressive models to non-stationary time series by minimum AIC procedure.

Usage
mlocar(y, max.order = NULL, span, const = 0, plot = TRUE)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>a univariate time series.</td>
</tr>
<tr>
<td>max.order</td>
<td>upper limit of the order of AR model. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.</td>
</tr>
<tr>
<td>span</td>
<td>length of the basic local span.</td>
</tr>
<tr>
<td>const</td>
<td>integer. 0 denotes constant vector is not included as a regressor and 1 denotes constant vector is included as the first regressor.</td>
</tr>
<tr>
<td>plot</td>
<td>logical. If TRUE (default), spectrums pspec are plotted.</td>
</tr>
</tbody>
</table>

Details

The data of length $n$ are divided into $k$ locally stationary spans,

$$| < --- n_1 --- > | < --- n_2 --- > | < --- n_3 --- > | ..... | < --- n_k --- > |$$

where $n_i$ ($i = 1, \ldots, k$) denotes the number of basic spans, each of length span, which constitute the $i$-th locally stationary span. At each local span, the process is represented by a stationary autoregressive model.

Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>mean.</td>
</tr>
<tr>
<td>var</td>
<td>variance.</td>
</tr>
<tr>
<td>ns</td>
<td>the number of local spans.</td>
</tr>
<tr>
<td>order</td>
<td>order of the current model.</td>
</tr>
<tr>
<td>arcoef</td>
<td>AR coefficients of current model.</td>
</tr>
<tr>
<td>v</td>
<td>innovation variance of the current model.</td>
</tr>
<tr>
<td>init</td>
<td>initial point of the data fitted to the current model.</td>
</tr>
<tr>
<td>end</td>
<td>end point of the data fitted to the current model.</td>
</tr>
<tr>
<td>pspec</td>
<td>power spectrum.</td>
</tr>
<tr>
<td>npre</td>
<td>data length of the preceding stationary block.</td>
</tr>
<tr>
<td>nnew</td>
<td>data length of the new block.</td>
</tr>
<tr>
<td>order.mov</td>
<td>order of the moving model.</td>
</tr>
<tr>
<td>v.mov</td>
<td>innovation variance of the moving model.</td>
</tr>
<tr>
<td>aic.mov</td>
<td>AIC of the moving model.</td>
</tr>
<tr>
<td>order.const</td>
<td>order of the constant model.</td>
</tr>
<tr>
<td>v.const</td>
<td>innovation variance of the constant model.</td>
</tr>
<tr>
<td>aic.const</td>
<td>AIC of the constant model.</td>
</tr>
</tbody>
</table>
References


Examples

```r
data(locarData)
z <- mlocar(locarData, max.order = 10, span = 300, const = 0)
z$arcoef
```

---

**mlomar**

*Minimum AIC Method of Locally Stationary Multivariate AR Model Fitting*

**Description**

Locally fit multivariate autoregressive models to non-stationary time series by the minimum AIC procedure using the householder transformation.

**Usage**

```r
mlomar(y, max.order = NULL, span, const = 0)
```

**Arguments**

- `y`: a multivariate time series.
- `max.order`: upper limit of the order of AR model, less than or equal to $n/2d$ where $n$ is the length and $d$ is the dimension of the time series $y$. Default is $\min(2\sqrt{n}, n/2d)$.
- `span`: length of basic local span. Let $m$ denote `max.order`, if $n - m - 1$ is less than or equal to span or $n - m - 1$ - span is less than $2md + const$, span is $n - m$.
- `const`: integer. '0' denotes constant vector is not included as a regressor and '1' denotes constant vector is included as the first regressor.

**Details**

The data of length $n$ are divided into $k$ locally stationary spans,

```
<--- n1 --- > | <--- n2 --- > | <--- n3 --- > | ..... | <--- nk --- > |
```

where $n_i$ ($i = 1, \ldots, k$) denoted the number of basic spans, each of length span, which constitute the $i$-th locally stationary span. At each local span, the process is represented by a stationary autoregressive model.
Value

- mean: mean.
- var: variance.
- ns: the number of local spans.
- order: order of the current model.
- aic: AIC of the current model.
- arcoef: AR coefficient matrices of the current model. \(\text{arcoef}[m][i,j,k]\) shows the value of \(i\)-th row, \(j\)-th column, \(k\)-th order of \(m\)-th model.
- v: innovation variance of the current model.
- init: initial point of the data fitted to the current model.
- end: end point of the data fitted to the current model.
- npre: data length of the preceding stationary block.
- nnew: data length of the new block.
- order.mov: order of the moving model.
- aic.mov: AIC of the moving model.
- order.const: order of the constant model.
- aic.const: AIC of the constant model.

References


Examples

data(Amerikamaru)
mlomar(Amerikamaru, max.order = 10, span = 300, const = 0)

mulbar

Multivariate Bayesian Method of AR Model Fitting

Description

Determine multivariate autoregressive models by a Bayesian procedure. The basic least squares estimates of the parameters are obtained by the householder transformation.

Usage

mulbar(y, max.order = NULL, plot = FALSE)
Arguments

- **y**: a multivariate time series.
- **max.order**: upper limit of the order of AR model, less than or equal to $n/2d$ where $n$ is the length and $d$ is the dimension of the time series $y$. Default is $\min(2\sqrt{n}, n/2d)$.
- **plot**: logical. If TRUE, daic is plotted.

Details

The statistic AIC is defined by

$$AIC = n \log(det(v)) + 2k,$$

where $n$ is the number of data, $v$ is the estimate of innovation variance matrix, $det$ is the determinant and $k$ is the number of free parameters.

Bayesian weight of the $m$-th order model is defined by

$$W(n) = const \times \frac{C(m)}{m+1},$$

where $const$ is the normalizing constant and $C(m) = \exp(-0.5AIC(m))$. The Bayesian estimates of partial autoregression coefficient matrices of forward and backward models are obtained by ($m = 1, \ldots, lag$)

$$G(m) = G(m)D(m),$$
$$H(m) = H(m)D(m),$$

where the original $G(m)$ and $H(m)$ are the (conditional) maximum likelihood estimates of the highest order coefficient matrices of forward and backward AR models of order $m$ and $D(m)$ is defined by

$$D(m) = W(m) + \ldots + W(lag).$$

The equivalent number of parameters for the Bayesian model is defined by

$$ek = \{D(1)^2 + \ldots + D(lag)^2\}id + \frac{id(id + 1)}{2}$$

where $id$ denotes dimension of the process.

Value

- **mean**: mean.
- **var**: variance.
- **v**: innovation variance.
- **aic**: AIC.
- **aicmin**: minimum AIC.
- **daic**: AIC-aicmin.
- **order.maice**: order of minimum AIC.
- **v.maice**: MAICE innovation variance.
mulcor

bweight  Bayesian weights.
integra.bweight integrated Bayesian Weights.

arcoef.for  AR coefficients (forward model). arcoef.for[i,j,k] shows the value of i-th row, j-th column, k-th order.
arcoef.back  AR coefficients (backward model). arcoef.back[i,j,k] shows the value of i-th row, j-th column, k-th order.
pacoef.for  partial autoregression coefficients (forward model).
pacoef.back  partial autoregression coefficients (backward model).
v.bay innovation variance of the Bayesian model.
aic.bay equivalent AIC of the Bayesian (forward) model.

References


Examples

data(Powerplant)
z <- mulbar(Powerplant, max.order = 10)
z$pacoef.for
z$pacoef.back

mulcor(y, lag = NULL, plot = TRUE, lag_axis = TRUE)

Description

Estimate multiple correlation.

Usage

mulcor(y, lag = NULL, plot = TRUE, lag_axis = TRUE)

Arguments

y a multivariate time series.
lag maximum lag. Default is $2\sqrt{n}$, where $n$ is the length of the time series y.
plot logical. If TRUE (default), correlations cor are plotted.
lag_axis logical. If TRUE (default) with plot=TRUE, x-axis is drawn.
Value

cov  covariances.
cor  correlations (normalized covariances).
mean  mean.

References


Examples

```r
# Example 1
y <- rnorm(1000)
dim(y) <- c(500,2)
mulcor(y, lag_axis = FALSE)

# Example 2
xorg <- rnorm(1003)
x <- matrix(0, nrow = 1000, ncol = 2)
x[, 1] <- xorg[1:1000]
x[, 2] <- xorg[4:1003] + 0.5*rnorm(1000)
mulcor(x, lag = 20)
```

---

**mulfrf**

*Frequency Response Function (Multiple Channel)*

Description

Compute multiple frequency response function, gain, phase, multiple coherency, partial coherency and relative error statistics.

Usage

```r
mulfrf(y, lag = NULL, iovar = NULL)
```

Arguments

- **y**  a multivariate time series.
- **lag**  maximum lag. Default is $2\sqrt{n}$, where $n$ is the number of rows in $y$.
- **iovar**  input variables $iovar[i]$ ($i = 1, k$) and output variable $iovar[k+1]$ ($1 \leq i \leq d$), where $d$ is the number of columns in $y$. Default is $c(1 : d)$. 
**Value**

- `cospec`: spectrum (complex).
- `freqr`: frequency response function: real part.
- `freqi`: frequency response function: imaginary part.
- `gain`: gain.
- `phase`: phase.
- `pcoh`: partial coherency.
- `errstat`: relative error statistics.
- `mcoh`: multiple coherency.

**References**


**Examples**

```r
ar <- array(0, dim = c(3,3,2))
ar[, , 1] <- matrix(c(0.4, 0, 0.3,
                     0.2, -0.1, -0.5,
                     0.3, 0.1, 0), nrow = 3, ncol = 3, byrow = TRUE)
ar[, , 2] <- matrix(c(0, -0.3, 0.5,
                     0.7, -0.4, 1,
                     0, -0.5, 0.3), nrow = 3, ncol = 3, byrow = TRUE)
x <- matrix(rnorm(200*3), nrow = 200, ncol = 3)
y <- mfilter(x, ar, "recursive")
mulfrf(y, lag = 20)
```

---

**mulmar**

*Multivariate Case of Minimum AIC Method of AR Model Fitting*

**Description**

Fit a multivariate autoregressive model by the minimum AIC procedure. Only the possibilities of zero coefficients at the beginning and end of the model are considered. The least squares estimates of the parameters are obtained by the householder transformation.

**Usage**

```r
mulmar(y, max.order = NULL, plot = FALSE)
```

**Arguments**

- `y`: a multivariate time series.
- `max.order`: upper limit of the order of AR model, less than or equal to \( n/2d \) where \( n \) is the length and \( d \) is the dimension of the time series \( y \). Default is \( \min(2\sqrt{n}, n/2d) \).
- `plot`: logical. If TRUE, `daic[[1]],...,daic[[d]]` are plotted.
Details

Multivariate autoregressive model is defined by

\[ y(t) = A(1)y(t - 1) + A(2)y(t - 2) + \ldots + A(p)y(t - p) + u(t), \]

where \( p \) is order of the model and \( u(t) \) is Gaussian white noise with mean 0 and variance matrix \( \text{matv} \). AIC is defined by

\[ AIC = n \log(\det(v)) + 2k, \]

where \( n \) is the number of data, \( v \) is the estimate of innovation variance matrix, \( \det \) is the determinant and \( k \) is the number of free parameters.

Value

- mean: mean.
- var: variance.
- v: innovation variance.
- aic: AIC.
- aicmin: minimum AIC.
- daic: AIC-aicmin.
- order.maice: order of minimum AIC.
- v.maice: MAICE innovation variance.
- np: number of parameters.
- jnd: specification of \( i \)-th regressor.
- subregcoef: subset regression coefficients.
- rvar: residual variance.
- aicf: final estimate of AIC (= \( n \log(rvar) + 2np \)).
- respns: instantaneous response.
- regcoef: regression coefficients matrix.
- matv: innovation variance matrix.
- morder: order of the MAICE model.
- arcoef: AR coefficients. \( \text{arcoef}[i,j,k] \) shows the value of \( i \)-th row, \( j \)-th column, \( k \)-th order.
- aicsum: the sum of aicf.

References


Examples

# Example 1
data(Powerplant)
z <- mulmar(Powerplant, max.order = 10)
z$arcoef

# Example 2
ar <- array(0, dim = c(3,3,2))
ar[, , 1] <- matrix(c(0.4, 0, 0.3, 0.2, -0.1, -0.5, 0.3, 0.1, 0), nrow = 3, ncol = 3, byrow = TRUE)
ar[, , 2] <- matrix(c(0, -0.3, 0.5, 0.7, -0.4, 1, 0, -0.5, 0.3), nrow = 3, ncol = 3, byrow = TRUE)
x <- matrix(rnorm(200*3), nrow = 200, ncol = 3)
y <- mfilter(x, ar, "recursive")
z <- mulmar(y, max.order = 10)
z$arcoef

mulnos Relative Power Contribution

Description

Compute relative power contributions in differential and integrated form, assuming the orthogonality between noise sources.

Usage

mulnos(y, max.order = NULL, control = NULL, manip = NULL, h)

Arguments

y          a multivariate time series.
max.order   upper limit of model order. Default is $2\sqrt{n}$, where $n$ is the length of time series $y$.
control     controlled variables. Default is $c(1 : d)$, where $d$ is the dimension of the time series $y$.
manip       manipulated variables. Default number of manipulated variable is '0'.
h           specify frequencies $i/2h$ ($i = 0, \ldots, h$).

Value

nperr       a normalized prediction error covariance matrix.
diffr       differential relative power contribution.
integr      integrated relative power contribution.
References


Examples

```r
ar <- array(0, dim = c(3,3,2))
ar[, , 1] <- matrix(c(0.4, 0, 0.3,
                    0.2, -0.1, -0.5,
                    0.3, 0.1, 0), nrow = 3, ncol = 3, byrow = TRUE)
ar[, , 2] <- matrix(c(0, -0.3, 0.5,
                    0.7, -0.4, 1,
                    0, -0.5, 0.3), nrow = 3, ncol = 3, byrow = TRUE)
x <- matrix(rnorm(200*3), nrow = 200, ncol = 3)
y <- mfilter(x, ar, "recursive")
mulnos(y, max.order = 10, h = 20)
```

**mulrsp**  
Multiple Rational Spectrum

---

**Description**

Compute rational spectrum for d-dimensional ARMA process.

**Usage**

```r
mulrsp(h, d, cov, ar = NULL, ma = NULL, log = FALSE, plot = TRUE, plot.scale = FALSE)
```

**Arguments**

- `h`: specify frequencies $i/2h$ ($i = 0, 1, ..., h$).
- `d`: dimension of the observation vector.
- `cov`: covariance matrix.
- `ar`: coefficient matrix of autoregressive model. $ar[i,j,k]$ shows the value of $i$-th row, $j$-th column, $k$-th order.
- `ma`: coefficient matrix of moving average model. $ma[i,j,k]$ shows the value of $i$-th row, $j$-th column, $k$-th order.
- `log`: logical. If TRUE, rational spectrums $rspec$ are plotted as $log(rspec)$.
- `plot`: logical. If TRUE, rational spectrums $rspec$ are plotted.
- `plot.scale`: logical. IF TRUE, the common range of the $y$-axis is used.

**Details**

ARMA process:

$$y(t) - A(1)y(t - 1) - ... - A(p)y(t - p) = u(t) - B(1)u(t - 1) - ... - B(q)u(t - q)$$

where $u(t)$ is a white noise with zero mean vector and covariance matrix $cov$. 
mulspe

Value
rspec rational spectrum.
scoh simple coherence.

References

Examples
# Example 1 for the normal distribution
xorg <- rnorm(1003)
x <- matrix(0, nrow = 1000, ncol = 2)
x[, 1] <- xorg[1:1000]
x[, 2] <- xorg[4:1003] + 0.5*rnorm(1000)
aaa <- ar(x)
mulrsp(h = 20, d = 2, cov = aaa$var.pred, ar = aaa$ar, plot = TRUE, plot.scale = TRUE)

# Example 2 for the AR model
ar <- array(0, dim = c(3,3,2))
ar[, , 1] <- matrix(c(0.4, 0, 0.3,
                      0.2, -0.1, -0.5,
                      0.3, 0.1, 0), nrow = 3, ncol = 3, byrow = TRUE)
ar[, , 2] <- matrix(c(0, -0.3, 0.5,
                      0.7, -0.4, 1,
                      0, -0.5, 0.3), nrow = 3, ncol = 3, byrow = TRUE)
x <- matrix(rnorm(200*3), nrow = 200, ncol = 3)
y <- mfilter(x, ar, "recursive")
z <- fpec(y, max.order = 10)
mulrsp(h = 20, d = 3, cov = z$perr, ar = z$arcoef)

mulspe

Multiple Spectrum

Description
Compute multiple spectrum estimates using Akaike window or Hanning window.

Usage
mulspe(y, lag = NULL, window = "Akaike", plot = TRUE, plot.scale = FALSE)
Arguments

- **y**
  a multivariate time series with \( d \) variables and \( n \) observations.

- **lag**
  maximum lag. Default is \( 2\sqrt{n} \), where \( n \) is the number of observations.

- **window**
  character string giving the definition of smoothing window. Allowed strings are "Akaike" (default) or "Hanning".

- **plot**
  logical. If TRUE (default) spectrums are plotted as \((d, d)\) matrix.
  - Diagonal parts : Auto spectrums for each series.
  - Lower triangular parts : Amplitude spectrums.
  - Upper triangular part : Phase spectrums.

- **plot.scale**
  logical. IF TRUE, the common range of the \( y \)-axis is used.

Details

Hanning Window : \[ a_1(0)=0.5, \quad a_1(1)=a_1(-1)=0.25, \quad a_1(2)=a_1(-2)=0 \]
Akaike Window : \[ a_2(0)=0.625, \quad a_2(1)=a_2(-1)=0.25, \quad a_2(2)=a_2(-2)=-0.0625 \]

Value

- **spec**
  spectrum smoothing by 'window'.
  - Lower triangular parts : Real parts
  - Upper triangular parts : Imaginary parts

- **stat**
  test statistics.

- **coh**
  simple coherence by 'window'.

References


Examples

```r
sgnl <- rnorm(1003)
x <- matrix(0, nrow = 1000, ncol = 2)
x[, 1] <- sgnl[4:1003]
x[, 2] <- 0.9*x[i-3,1] + 0.2*N(0,1)
x[, 2] <- 0.9*sgnl[1:1000] + 0.2*rnorm(1000)
mulspe(x, lag = 100, window = "Hanning", plot.scale = TRUE)
```
Description

Locally fit autoregressive models to non-stationary time series by AIC criterion.

Usage

`nonst(y, span, max.order = NULL, plot = TRUE)`

Arguments

- `y`: a univariate time series.
- `span`: length of the basic local span.
- `max.order`: highest order of AR model. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.
- `plot`: logical. If TRUE (the default), spectrums are plotted.

Details

The basic AR model is given by

\[ y(t) = A(1)y(t - 1) + A(2)y(t - 2) + \ldots + A(p)y(t - p) + u(t), \]

where $p$ is order of the AR model and $u(t)$ is innovation variance. AIC is defined by

\[ AIC = n \log(\text{det}(sd)) + 2k, \]

where $n$ is the length of data, $sd$ is the estimates of the innovation variance and $k$ is the number of parameter.

Value

- `ns`: the number of local spans.
- `arcoef`: AR coefficients.
- `v`: innovation variance.
- `aic`: AIC.
- `daic21` = AIC2 - AIC1.
- `daic` = daic21/n (n is the length of the current model).
- `init`: start point of the data fitted to the current model.
- `end`: end point of the data fitted to the current model.
- `pspec`: power spectrum.
References


Examples

```r
# Non-stationary Test Data
data(nonstData)
nonst(nonstData, span = 700, max.order = 49)
```

<table>
<thead>
<tr>
<th>nonstData</th>
<th>Non-stationary Test Data</th>
</tr>
</thead>
</table>

Description

A non-stationary data for testing `nonst`.

Usage

```r
data(nonstData)
```

Format

A time series of 2100 observations.

Source


<table>
<thead>
<tr>
<th>optdes</th>
<th>Optimal Controller Design</th>
</tr>
</thead>
</table>

Description

Compute optimal controller gain matrix for a quadratic criterion defined by two positive definite matrices Q and R.

Usage

```r
optdes(y, max.order = NULL, ns, q, r)
```
Arguments

y  
a multivariate time series.

max.order  
upper limit of model order. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.

ns  
number of D.P. stages.

q  
positive definite $(m, m)$ matrix $Q$, where $m$ is the number of controlled variables. A quadratic criterion is defined by $Q$ and $R$.

r  
positive definite $(l, l)$ matrix $R$, where $l$ is the number of manipulated variables.

Value

perr  
prediction error covariance matrix.

trans  
first $m$ columns of transition matrix, where $m$ is the number of controlled variables.

gamma  
gamma matrix.

gain  
gain matrix.

References


Examples

# Multivariate Example Data
ar <- array(0, dim = c(3,3,2))
ar[, , 1] <- matrix(c(0.4, 0, 0.3,
                    0.2, -0.1, -0.5,
                    0.3, 0.1, 0), nrow= 3, ncol= 3, byrow = TRUE)
ar[, , 2] <- matrix(c(0, -0.3, 0.5,
                    0.7, -0.4, 1,
                    0, -0.5, 0.3), nrow= 3, ncol= 3, byrow = TRUE)
x <- matrix(rnorm(200*3), nrow = 200, ncol = 3)
y <- mfilter(x, ar, "recursive")
q.mat <- matrix(c(0.16,0,0,0.09), nrow = 2, ncol = 2)
r.mat <- as.matrix(0.001)
optdes(y, ns = 20, q = q.mat, r = r.mat)

Description

Perform optimal control simulation and evaluate the means and variances of the controlled and manipulated variables X and Y.
Usage

```r
optsim(y, max.order = NULL, ns, q, r, noise = NULL, len, plot = TRUE)
```

Arguments

- `y`: a multivariate time series.
- `max.order`: upper limit of model order. Default is \(2\sqrt{n}\), where \(n\) is the length of the time series `y`.
- `ns`: number of steps of simulation.
- `q`: positive definite matrix \(Q\).
- `r`: positive definite matrix \(R\).
- `noise`: noise. If not provided, Gaussian vector white noise with the length `len` is generated.
- `len`: length of white noise record.
- `plot`: logical. If `TRUE` (default), controlled variables \(X\) and manipulated variables \(Y\) are plotted.

Value

- `trans`: first \(m\) columns of transition matrix, where \(m\) is the number of controlled variables.
- `gamma`: gamma matrix.
- `gain`: gain matrix.
- `convar`: controlled variables \(X\).
- `manvar`: manipulated variables \(Y\).
- `xmean`: mean of \(X\).
- `ymean`: mean of \(Y\).
- `xvar`: variance of \(X\).
- `yvar`: variance of \(Y\).
- `x2sum`: sum of \(X^2\).
- `y2sum`: sum of \(Y^2\).
- `x2mean`: mean of \(X^2\).
- `y2mean`: mean of \(Y^2\).

References

Examples

```r
# Multivariate Example Data
ar <- array(0, dim = c(3,3,2))
ar[, , 1] <- matrix(c(0.4, 0, 0.3,
                      0.2, -0.1, -0.5,
                      0.3, 0.1, 0), nrow = 3, ncol = 3, byrow = TRUE)
ar[, , 2] <- matrix(c(0, -0.3, 0.5,
                      0.7, -0.4, 1,
                      0, -0.5, 0.3), nrow = 3, ncol = 3, byrow = TRUE)
x <- matrix(rnorm(200*3), nrow = 200, ncol = 3)
y <- mfilter(x, ar, "recursive")
q.mat <- matrix(c(0.16,0,0,0.09), nrow = 2, ncol = 2)
r.mat <- as.matrix(0.001)
optsim(y, max.order = 10, ns = 20, q = q.mat, r = r.mat, len = 20)
```

---

**perars**  
*Periodic Autoregression for a Scalar Time Series*

Description

This is the program for the fitting of periodic autoregressive models by the method of least squares realized through householder transformation.

Usage

```r
perars(y, ni, lag = NULL, ksw = 0)
```

Arguments

- `y`: a univariate time series.
- `ni`: number of instants in one period.
- `lag`: maximum lag of periods. Default is \(2\sqrt{ni}\).
- `ksw`: integer. '0' denotes constant vector is not included as a regressor and '1' denotes constant vector is included as the first regressor.

Details

Periodic autoregressive model \((i = 1, \ldots, nd, j = 1, \ldots, ni)\) is defined by

\[
z(i, j) = y(ni(i - 1) + j),
\]

\[
z(i, j) = c(j) + A(1, j, 0)z(i, 1) + \ldots + A(j - 1, j, 0)z(i, j - 1) + A(1, j, 1)z(i - 1, 1) + \ldots + A(ni, j, 1)z(i - 1, ni) + \ldots + u(i, j),
\]

where \(nd\) is the number of periods, \(ni\) is the number of instants in one period and \(u(i, j)\) is the Gaussian white noise. When ksw is set to '0', the constant term \(c(j)\) is excluded.

The statistics AIC is defined by \(AIC = n \log(det(v)) + 2k\), where \(n\) is the length of data, \(v\) is the estimate of the innovation variance matrix and \(k\) is the number of parameters. The outputs are the estimates of the regression coefficients and innovation variance of the periodic AR model for each instant.
Value

- **mean**: mean.
- **var**: variance.
- **subset**: specification of $i$-th regressor ($i = 1, \ldots, ni$).
- **regcoef**: regression coefficients.
- **rvar**: residual variances.
- **np**: number of parameters.
- **aic**: AIC.
- **v**: innovation variance matrix.
- **arcoef**: AR coefficient matrices. $\text{arcoef}[i,,k]$ shows $i$-th regressand of $k$-th period former.
- **const**: constant vector.
- **morder**: order of the MAICE model.

References


Examples

```r
data(Airpollution)
perars(Airpollution, ni = 6, lag = 2, ksw = 1)
```

Description

A Power plant data for testing `mulbar` and `mulmar`.

Usage

```r
data(Powerplant)
```

Format

A 2-dimensional array with 500 observations on 3 variables.

```
[ , 1] command
[ , 2] temperature
[ , 3] fuel
```
**prdctr**

**Description**
Operate on a real record of a vector process and compute predicted values.

**Usage**

```r
prdctr(y, r, s, h, arcoef, macoef = NULL, impulse = NULL, v, plot = TRUE)
```

**Arguments**

- `y`: a univariate time series or a multivariate time series.
- `r`: one step ahead prediction starting position $R$.
- `s`: long range forecast starting position $S$.
- `h`: maximum span of long range forecast $H$.
- `arcoef`: AR coefficient matrices.
- `macoef`: MA coefficient matrices.
- `impulse`: impulse response matrices.
- `v`: innovation variance.
- `plot`: logical. If TRUE (default), the real data and predicted values are plotted.

**Details**

One step ahead Prediction starts at time $R$ and ends at time $S$. Prediction is continued without new observations until time $S + H$. Basic model is the autoregressive moving average model of $y(t)$ which is given by

$$y(t) - A(t)y(t-1) - ... - A(p)y(t-p) = u(t) - B(1)u(t-1) - ... - B(q)u(t-q),$$

where $p$ is AR order and $q$ is MA order.

**Value**

- `predct`: predicted values: `predct[i]` ($r \leq i \leq s+h$).
- `ys`: `predct[i] - y[i]` ($r \leq i \leq n$).
- `pstd`: `predct[i] + (standard deviation)` ($s \leq i \leq s+h$).
- `p2std`: `predct[i] + 2*(standard deviation)` ($s \leq i \leq s+h$).
- `p3std`: `predct[i] + 3*(standard deviation)` ($s \leq i \leq s+h$).
- `mstd`: `predct[i] - (standard deviation)` ($s \leq i \leq s+h$).
- `m2std`: `predct[i] - 2*(standard deviation)` ($s \leq i \leq s+h$).
- `m3std`: `predct[i] - 3*(standard deviation)` ($s \leq i \leq s+h$).
References


Examples

```r
# "arima.sim" is a function in "stats".
# Note that the sign of MA coefficient is opposite from that in "timsac".
y <- arima.sim(list(order=c(2,0,1), ar=c(0.64,-0.8), ma=c(-0.5)), n = 1000)
y1 <- y[1:900]
z <- autoarmafit(y1)
ar <- z$model[[1]]$arcoef
ma <- z$model[[1]]$macoef
var <- z$model[[1]]$v
y2 <- y[901:990]
prdctr(y2, r = 50, s = 90, h = 10, arcoef = ar, macoef = ma, v = var)
```

---

raspec | Rational Spectrum

Description

Compute power spectrum of ARMA process.

Usage

```r
raspec(h, var, arcoef = NULL, macoef = NULL, log = FALSE, plot = TRUE)
```

Arguments

- **h**: specify frequencies $i/2h$ ($i = 0, 1, \ldots, h$).
- **var**: variance.
- **arcoef**: AR coefficients.
- **macoef**: MA coefficients.
- **log**: logical. If TRUE, the spectrum is plotted as log(raspec).
- **plot**: logical. If TRUE (default), the spectrum is plotted.

Details

ARMA process:

$$y(t) - a(1)y(t-1) - \ldots - a(p)y(t-p) = u(t) - b(1)u(t-1) - \ldots - b(q)u(t-q)$$

where $p$ is AR order, $q$ is MA order and $u(t)$ is a white noise with zero mean and variance equal to var.
**Value**

raspec gives the rational spectrum.

**References**


**Examples**

```r
# Example 1 for the AR model
raspec(h = 100, var = 1, arcoef = c(0.64, -0.8))

# Example 2 for the MA model
raspec(h = 20, var = 1, macoef = c(0.64, -0.8))
```

---

### sglfre

**Frequency Response Function (Single Channel)**

**Description**

Compute 1-input, 1-output frequency response function, gain, phase, coherency and relative error statistics.

**Usage**

```r
sglfre(y, lag = NULL, invar, outvar)
```

**Arguments**

- **y**: a multivariate time series.
- **lag**: maximum lag. Default $2\sqrt{n}$, where $n$ is the length of the time series y.
- **invar**: within $d$ variables of the spectrum, invar-th variable is taken as an input variable.
- **outvar**: within $d$ variables of the spectrum, outvar-th variable is taken as an output variable.

**Value**

- **inspec**: power spectrum (input).
- **outspec**: power spectrum (output).
- **cspec**: co-spectrum.
- **qspec**: quad-spectrum.
- **gain**: gain.
- **coh**: coherency.
freqr  frequency response function: real part.
freqi  frequency response function: imaginary part.
errstat relative error statistics.
phase  phase.

References

Examples

```r
ar <- array(0, dim = c(3,3,2))
ar[, , 1] <- matrix(c(0.4, 0, 0.3,
                     0.2, -0.1, -0.5,
                     0.3, 0.1, 0), nrow = 3, ncol = 3, byrow = TRUE)
ar[, , 2] <- matrix(c(0, -0.3, 0.5,
                     0.7, -0.4, 1,
                     0, -0.5, 0.3), nrow = 3, ncol = 3, byrow = TRUE)
x <- matrix(rnorm(200*3), nrow = 200, ncol = 3)
y <- mfilter(x, ar, "recursive")
sglfre(y, lag = 20, invar = 1, outvar = 2)
```

---

**simcon**  

*Optimal Controller Design and Simulation*

**Description**

Produce optimal controller gain and simulate the controlled process.

**Usage**

```r
simcon(span, len, r, arcoef, impulse, v, weight)
```

**Arguments**

- `span` span of control performance evaluation.
- `len` length of experimental observation.
- `r` dimension of control input, less than or equal to `d` (dimension of a vector).
- `arcoef` matrices of autoregressive coefficients. `arcoef[i,j,k]` shows the value of `i`-th row, `j`-th column, `k`-th order.
- `impulse` impulse response matrices.
- `v` covariance matrix of innovation.
- `weight` weighting matrix of performance.
Details

The basic state space model is obtained from the autoregressive moving average model of a vector process \( y(t) \);

\[
y(t) - A(1)y(t-1) - \ldots - A(p)y(t-p) = u(t) - B(1)u(t-1) - \ldots - B(p-1)u(t-p+1),
\]

where \( A(i) \) \( (i = 1, \ldots, p) \) are the autoregressive coefficients of the ARMA representation of \( y(t) \).

Value

- **gain**: controller gain.
- **ave**: average value of \( i \)-th component of \( y \).
- **var**: variance.
- **std**: standard deviation.
- **bc**: sub matrices \((pd, r)\) of impulse response matrices, where \( p \) is the order of the process, \( d \) is the dimension of the vector and \( r \) is the dimension of the control input.
- **bd**: sub matrices \((pd, d - r)\) of impulse response matrices.

References


Examples

```r
x <- matrix(rnorm(1000*2), nrow = 1000, ncol = 2)
ma <- array(0, dim = c(2,2,2))
ma[, , 1] <- matrix(c( -1.0, 0.0,
0.0, -1.0), nrow = 2, ncol = 2, byrow = TRUE)
ma[, , 2] <- matrix(c( -0.2, 0.0,
-0.1, -0.3), nrow = 2, ncol = 2, byrow = TRUE)
y <- mfilter(x, ma, "convolution")

ar <- array(0, dim = c(2,2,3))
ar[, , 1] <- matrix(c( -1.0, 0.0,
0.0, -1.0), nrow = 2, ncol = 2, byrow = TRUE)
ar[, , 2] <- matrix(c( -0.5, -0.2,
-0.2, -0.5), nrow = 2, ncol = 2, byrow = TRUE)
ar[, , 3] <- matrix(c( -0.3, -0.05,
-0.1, -0.3), nrow = 2, ncol = 2, byrow = TRUE)
y <- mfilter(y, ar, "recursive")

z <- markov(y)
weight <- matrix(c(0.0002, 0.0,
0.0, 2.9), nrow = 2, ncol = 2, byrow = TRUE)
simcon(span = 50, len = 700, r = 1, z$arcoef, z$impulse, z$v, weight)
```
thirmo

Third Order Moments

Description

Compute the third order moments.

Usage

thirmo(y, lag = NULL, plot = TRUE)

Arguments

- **y**: a univariate time series.
- **lag**: maximum lag. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.
- **plot**: logical. If TRUE (default), autocovariance acor is plotted.

Value

- **mean**: mean.
- **acov**: autocovariance.
- **acor**: normalized covariance.
- **tmomnt**: third order moments.

References


Examples

data(bispecData)
z <- thirmo(bispecData, lag = 30)
z$tmomnt
unibar

Univariate Bayesian Method of AR Model Fitting

Description

This program fits an autoregressive model by a Bayesian procedure. The least squares estimates of the parameters are obtained by the householder transformation.

Usage

unibar(y, ar.order = NULL, plot = TRUE)

Arguments

y a univariate time series.

ar.order order of the AR model. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.

plot logical. If TRUE (default), daic, pacof and pspec are plotted.

Details

The AR model is given by

$$y(t) = a(1)y(t - 1) + \ldots + a(p)y(t - p) + u(t),$$

where $p$ is AR order and $u(t)$ is Gaussian white noise with mean 0 and variance $v(p)$. The basic statistic AIC is defined by

$$AIC = n \log(\det(v)) + 2m,$$

where $n$ is the length of data, $v$ is the estimate of innovation variance, and $m$ is the order of the model.

Bayesian weight of the $m$-th order model is defined by

$$W(m) = CONST \times \frac{C(m)}{m + 1},$$

where $CONST$ is the normalizing constant and $C(m) = \exp(-0.5AIC(m))$. The equivalent number of free parameter for the Bayesian model is defined by

$$ek = D(1)^2 + \ldots + D(k)^2 + 1,$$

where $D(j)$ is defined by $D(j) = W(j) + \ldots + W(k)$. $m$ in the definition of AIC is replaced by $ek$ to define an equivalent AIC for a Bayesian model.
unibar

Value

mean  mean.
var   variance.
v     innovation variance.
aic   AIC.
aicmin minimum AIC.
daic  AIC-aicmin.
order.maice order of minimum AIC.
v.maice innovation variance attained at m=order.maice.
pacoef partial autocorrelation coefficients (least squares estimate).
bweight Bayesian Weight.
integra.bweight integrated Bayesian weights.
v.bay innovation variance of Bayesian model.
aic.bay AIC of Bayesian model.
np    equivalent number of parameters.
pacoef.bay partial autocorrelation coefficients of Bayesian model.
arcoef AR coefficients of Bayesian model.
pspec power spectrum.

References


Examples

data(Canadianlynx)
z <- unibar(Canadianlynx, ar.order = 20)
z$arcoef
unimar

Univariate Case of Minimum AIC Method of AR Model Fitting

Description
This is the basic program for the fitting of autoregressive models of successively higher by the method of least squares realized through householder transformation.

Usage
unimar(y, max.order = NULL, plot = FALSE)

Arguments
- **y**: a univariate time series.
- **max.order**: upper limit of AR order. Default is $2\sqrt{n}$, where $n$ is the length of the time series $y$.
- **plot**: logical. If TRUE, daic is plotted.

Details
The AR model is given by
$$y(t) = a(1)y(t - 1) + \ldots + a(p)y(t - p) + u(t),$$
where $p$ is AR order and $u(t)$ is Gaussian white noise with mean 0 and variance $v$. AIC is defined by
$$AIC = n \log(\det(v)) + 2k,$$
where $n$ is the length of data, $v$ is the estimates of the innovation variance and $k$ is the number of parameter.

Value
- **mean**: mean.
- **var**: variance.
- **v**: innovation variance.
- **aic**: AIC.
- **aicmin**: minimum AIC.
- **daic**: AIC-aicmin.
- **order.maice**: order of minimum AIC.
- **v.maice**: innovation variance attained at order.maice.
- **arcoef**: AR coefficients.
References


Examples

data(Canadianlynx)

z <- unimar(Canadianlynx, max.order = 20)
z$arcoef

wnoise

White Noise Generator

Description

Generate approximately Gaussian vector white noise.

Usage

wnoise(len, perr, plot = TRUE)

Arguments

len length of white noise record.
perr prediction error.
plot logical. If TRUE (default), white noises are plotted.

Value

wnoise gives white noises.

References


Examples

# Example 1
wnoise(len = 100, perr = 1)

# Example 2
v <- matrix(c(1, 0, 0,
              0, 2, 0,
              0, 0, 3), nrow = 3, ncol = 3, byrow = TRUE)
wnoise(len = 20, perr = v)
Exact Maximum Likelihood Method of Scalar ARMA Model Fitting

Description

Produce exact maximum likelihood estimates of the parameters of a scalar ARMA model.

Usage

xsarma(y, arcoefi, macoefi)

Arguments

- **y**: a univariate time series.
- **arcoefi**: initial estimates of AR coefficients.
- **macoefi**: initial estimates of MA coefficients.

Details

The ARMA model is given by

\[ y(t) - a(1)y(t-1) - \ldots - a(p)y(t-p) = u(t) - b(1)u(t-1) - \ldots - b(q)u(t-q), \]

where \( p \) is AR order, \( q \) is MA order and \( u(t) \) is a zero mean white noise.

Value

- **gradi**: initial gradient.
- **lkhoodi**: initial (-2)log likelihood.
- **arcoef**: final estimates of AR coefficients.
- **macoef**: final estimates of MA coefficients.
- **grad**: final gradient.
- **alph.ar**: final ALPH (AR part) at subroutine ARCHCK.
- **alph.ma**: final ALPH (MA part) at subroutine ARCHCK.
- **lkhood**: final (-2)log likelihood.
- **wnoise.var**: white noise variance.

References

Examples

# "arima.sim" is a function in "stats".
# Note that the sign of MA coefficient is opposite from that in "timsac".
arcoef <- c(1.45, -0.9)
macoef <- c(-0.5)
y <- arima.sim(list(order=c(2,0,1), ar=arcoef, ma=macoef), n = 100)
arcoefi <- c(1.5, -0.8)
mae <- c(0.0)
z <- xsarma(y, arcoefi, ma<-c)
z$arcoef
z$mae
## Index

* **datasets**
  - Airpollution, 3
  - Amerikamaru, 3
  - bispecData, 11
  - Blsallfood, 14
  - Canadianlynx, 17
  - LaborData, 28
  - locarData, 28
  - nonstData, 45
  - Powerplant, 50

* **package**
  - timsac-package, 2

* **ts**
  - armafit, 4
  - auspec, 5
  - autcor, 6
  - autoarmafit, 7
  - baysea, 8
  - bispec, 10
  - blocar, 11
  - blomar, 13
  - bsubst, 15
  - canarm, 17
  - canoca, 19
  - covgen, 20
  - decomp, 21
  - exsar, 23
  - fftcor, 24
  - fpeaut, 25
  - fpec, 27
  - markov, 29
  - mfilter, 30
  - mlocar, 32
  - mlomar, 33
  - mulbar, 35
  - mulcor, 37
  - mulfrf, 38
  - mulmar, 39
  - mulnos, 40
  - mulspe, 43
  - nonst, 44
  - optdes, 46
  - optsim, 47
  - perars, 48
  - prdctr, 50
  - raspec, 51
  - sglfre, 53
  - simcon, 54
  - thirmo, 55
  - unibar, 56
  - unimar, 58
  - wnoise, 59
  - xsarma, 60
  - Airpollution, 3
  - Amerikamaru, 3
  - arima.sim, 31
  - armafit, 4
  - auspec, 5
  - autcor, 6
  - autoarmafit, 7
  - baysea, 8
  - bispec, 10, 11
  - bispecData, 11
  - blocar, 11, 28
  - blomar, 3, 13
  - Blsallfood, 14
  - bsubst, 15, 17
  - Canadianlynx, 17
  - canarm, 17
  - canoca, 19
  - convolve, 31
  - covgen, 20
  - decomp, 21
  - exsar, 17, 23

63
fftcor, 24
fpeaut, 25
fpec, 27

LaborData, 28
locarData, 28

markov, 29
mfilter, 30
mlocar, 28, 32
mlomar, 33
mulbar, 35, 50
mulcor, 37
mulfrf, 38
mulmar, 39, 50
mulnos, 40
mulrsp, 41
mulspe, 43

nonst, 44, 45
nonstData, 45

optdes, 46
optsim, 47

perars, 3, 48
Powerplant, 50
prdctr, 50
print.autoarmafit(autoarmafit), 7
print.blomar(blomar), 13
print.fpec(fpec), 27
print.mlomar(mlomar), 33
print.mulcor(mulcor), 37
print.perars(perars), 48
print.prdctr(prdctr), 50
ptint.mulspe(mulspe), 43

raspec, 51
sglfre, 53
simcon, 29, 54

thirmo, 11, 55
timsac(timsac-package), 2
timsac-package, 2

unibar, 17, 56
unimar, 17, 58

wnoise, 59
xsarma, 60