Package ‘timsac’

January 2, 2012

Version 1.2.1

Title TIMe Series Analysis and Control package

Author The Institute of Statistical Mathematics

Maintainer The Institute of Statistical Mathematics <`timsac@jasp.ism.ac.jp`>

Depends R (>= 2.4.0)

Description Time series analysis and control program package TIMSAC.

License GPL (>= 2)

Repository CRAN

Date/Publication 2009-03-21 08:16:26

R topics documented:

`Airpolution` *Airpolution Data*

Description

An air polution test data for "perars".

Usage

```
data(Airpolution)
```

Source

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

`Amerikamaru` *Amerikamaru Data*

Description

A multivariate non-stationary test data.

Usage

```
data(Amerikamaru)
```

Source

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

`armafit` *ARMA Model Fitting*

Description

Fit an ARMA model with specified order by using DAVIDON's algorithm.

Usage

```
armafit(y, model.order, tmp.file=NULL)
```

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.
`tmp.file`	a character string naming a file written intermediate results of model fitting. If NULL (default) output no file.

Details

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

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

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

Value

`arcoef`	maximum likelihood estimates of AR coefficients.
`macoef`	maximum likelihood estimates of MA coefficients.
`arstd`	standard deviation (AR).
`mastd`	standard deviation (MA).
`v`	innovation variance.
`aic`	AIC.
`grad`	final gradient.

References

H.Akaike, E.Arahata and T.Ozaki (1975) *Computer Science Monograph, No.5, Timsac74, A Time Series Analysis and Control Program Package (1).* The Institute of Statistical Mathematics.

Examples

```
# "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)
z <- armafit(y, model.order=c(2,1))
z$arcoef
z$macoef
```

`armaimp` *Caluculate Characteristics of Scalar ARMA Model*

Description

Caluculate impulse, autocovariance, partial autocorrelation function and characteristic roots of scalar ARMA model for given AR and MA coefficients.

Usage

```
armaimp( arcoef, macoef, v, n=1000, lag=NULL, nf=200, plot=TRUE )
```

Arguments

`arcoef`	AR coefficients.
`macoef`	MA coefficients.
`v`	innivation variance.
`n`	data length.
`lag`	maximum lag of autocovariance function. Default is 2*sqrt(n).
`nf`	number of frequencies in evaluating spectrum.
`plot`	logical. If TRUE (default) impulse response function, autocovariance, power spectrum and characteristic roos are plotted.

Details

The ARMA model is given by

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

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

Value

`impuls`	impulse response function.
`acov`	autocovariance function.
`parcor`	partial autocorrelation function.
`spec`	power spectrum.
`croot.ar`	characteristic roots of AR operator. Chracteristic root is a list with components named real(real part R), image(imaginary part I), amp(=sqrt(R**2+I**2)), atan(=ARCTAN(I/R)) and degree.
`croot.ma`	characteristic roots of MA operator.

References

G.Kitagawa (1993) *Time series analysis programing (in Japanese).* The Iwanami Computer Science Senes.

Examples

```
# ARMA model : y(n) = 0.9sqrt(3)y(n-1) - 0.81y(n-2) + v(n) -0.9sqrt(2)v(n-1) + 0.81v(n-2)
a <- c(0.9*sqrt(3), -0.81)
b <- c(0.9*sqrt(2), -0.81)
z <- armaimp( arcoef=a, macoef=b, v=1.0, n=1000, lag=20 )
z$croot.ar
z$croot.ma

# AR model : y(n) = 0.9sqrt(3)y(n-1) - 0.81y(n-2) + v(n)
z <- armaimp( arcoef=a, v=1.0, n=1000, lag=20 )
z$croot.ar

# MA model : y(n) = v(n) -0.9sqrt(2)v(n-1) + 0.81v(n-2)
z <- armaimp( macoef=b, v=1.0, n=1000, lag=20 )
z$croot.ma
```

`auspec` *Power Spectrum*

Description

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

Usage

```
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 values are "Akaike" (default) or "Hanning".
`log`	logical. If TRUE, the spectrum spec is plotted as log(spec).
`plot`	logical. If TRUE (default) the spectrum is plotted.

Details

Hanning Window : a1(0)=0.5, a1(1)=a1(-1)=0.25, a1(2)=a1(-2)=0

Akaike Window : a2(0)=0.625, a2(1)=a2(-1)=0.25, a2(2)=a2(-2)=-0.0625

Value

`spec`	spectrum smoothing by "window"
`stat`	test statistics.

References

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

Examples

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

`autcor` *Autocorrelation*

Description

Estimate autocovariances and autocorrelations.

Usage

```
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.

References

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

Examples

```
  # 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

```
autoarmafit(y, max.order=NULL, tmp.file=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.
`tmp.file`	a character string naming a file written intermediate results of model fitting. If NULL (default) output no file.

Details

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

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

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 zwro mean white noise. Pure autoregression is not allowed.

Value

`best.order`	the order of the best ARMA model.
`best.model`	Tte best choice of ARMA coefficients.
`model`	a list with components named `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` (AIC $= nlog(det(v)) + 2(p+q)$) and `grad` (Final gradient) in AIC increasing order.

References

H.Akaike, E.Arahata and T.Ozaki (1975) *Computer Science Monograph, No.5, Timsac74, A Time Series Analysis and Control Program Package (1).* The Institute of Statistical Mathematics.

Examples

```
# "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)
z <- autoarmafit(y)
z$best.order
z$best.model
```

`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, o
     rigid=1, zersum=1, delta=7, alpha=0.01, beta=0.01, gamma=0.1, spec=TRUE, plot=TRUE, separate.gra
```

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 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

$ABIC = -2(logmaximumlikelihoodofthemodel)$

Smaller value of ABIC represents better fit.

Value

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

References

H.Akaike, T.Ozaki, M.Ishiguro, Y.Ogata, G.Kitagawa, Y-H.Tamura, E.Arahata, K.Katsura and Y.Tamura (1985) *Computer Science Monograph, No.22, Timsac84 Part 1.* The Institute of Statistical Mathematics.

Examples

```
data(LaborData)
baysea(y=LaborData, forecast=12)
```

`bispec` *Bispectrum*

Description

Compute bi-spectrum using the direct Fourier transform of sample third order monents.

Usage

```
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 values 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 : a1(0)=0.5, a1(1)=a1(-1)=0.25, a1(2)=a1(-2)=0

Akaike Window : a2(0)=0.625, a2(1)=a2(-1)=0.25, a2(2)=a2(-2)=-0.0625

Value

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

References

H.Akaike, E.Arahata and T.Ozaki (1975) *Computer Science Monograph, No.6, Timsac74, A Time Series Analysis and Control Program Package (2).* The Institute of Statistical Mathematics.

Examples

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

`bispecData` *Univariate Test Data*

Description

A test data for "bispec" and "thirmo".

Usage

```
data(bispecData)
```

Source

H.Akaike, E.Arahata and T.Ozaki (1976) *Computer Science Monograph, No.6, Timsac74 A Time Series Analysis and Control Program Package (2).* The Institute of Statistical Mathematics.

`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.
`plot`	logical. If TRUE (default) spectrums pspec are plotted.

Details

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

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

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

$AIC = nslog(det(v)) + 2k,$

where ns is the length of new data, v is the 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

$AIC(j+1) = nslog(det(v(j)) + 2$

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

Value

`var`	variance.
`aic`	AIC.
`bweight`	Bayesian weight.
`pacoef`	partial autocorrelation.
`arcoef`	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

G.Kitagawa and H.Akaike (1978) A Procedure for The Modeling of Non-Stationary Time Series. Ann. Inst. Statist. Math., 30, B, 351–363.

H.Akaike (1978) A Bayesian Extension of the Minimin MIC Procedure of Autoregressive Model Fitting. Reseach Meno. NO.126. The Institute of The Statistical Mathematics.

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

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. Default is 2*sqrt(n), where n is the length of the time series y.
`span`	length of basic local span.

Details

The basic AR model is given by

$y(t) = A(1)y(t-1) + A(2)y(t-2) + ... + A(p)y(t-p) + u(t),$

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

Value

`mean`	mean.
`var`	variance.
`bweight`	Bayesian weight.
`aic`	AIC with respect to the present data.
`arcoef`	AR coefficients. 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

G.Kitagawa and H.Akaike (1978) A Procedure for the Modeling of Non-stationary Time Series. Ann. Inst. Statist. Math., 30, B, 351–363.

H.Akaike (1978) A Bayesian Extension of The Minimum AIC Procedure of Autoregressive Model Fitting. Research Memo. NO.126. The institute of Statistical Mathematics.

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

Examples

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

`Blsallfood` *Blsallfood data*

Description

A blsallfood test data for "decomp".

Usage

```
data(Blsallfood)
```

Source

H.Akaike, T.Ozaki, M.Ishiguro, Y.ogata, G.Kitagawa, Y-H.tamura, E.Arahata, K.Katsura and Y.tamura (1984) *Computer Science Monographs, Timsac-84 Part 1.* The Institute of Statistical Mathematics.

`bsubst` *Bayesian Type All Subset Analysis*

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

```
bsubst(y,mtype,lag=NULL,nreg,reg=NULL,term.lag=NULL,cstep=5,plot=TRUE)
```

Arguments

`y`	a univariate time series.
`mtype`	model type. Allowed values are `1` : (autoregressive model), `2` : (polinomial type non-linear model, lag's read in), `3` : (polinomial type non-linear model, lag's automatically set) and `4` : (AR-model with polinomial mean value function). `5`,`6` and `7` are originaly defined but omitted here.
`lag`	maximum time lag. Default is 2*sqrt(n), where `n` is the length of the time series `y`.

`nreg`	number of regressors.
`reg`	specification of regressor (mtype = 2). i-th regressor is defined by $z(n - L1(i)) * z(n - L2(i)) * z(n - L3(i))$, where L1(i) is reg(1,i), L2(i) is reg(2,i) and L3(i) is reg(3,i). 0-lag term z(n-0) is replaced by the constant 1.
`term.lag`	maximum time lag specify the regressors (L1(i),L2(i),L3(i)) (i=1,...,nreg) (mtype = 3). i-th regressor is defined by $z(n - L1(i)) * z(n - L2(i)) * z(n - L3(i))$, where 0-lag term z(n-0) is replaced by the constant 1. 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 cross term 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.
`plot`	logical. If TRUE (default) daic, pre.err and peautcor are plotted.

Details

The AR model is given by (mtype = 2)

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

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

$y(t) = a(1)z(t, 1) + a(2)z(t, 2) + ... + a(p)z(t, p) + u(t).$

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,...,nreg).
`aicmin`	minimum AIC.
`daic`	AIC(m)-aicmin (m=0,...,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 (normalisedby 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).
`perr`	prediction error.
`mean`	mean.
`var`	variance.
`skew`	skewness.
`peak`	peakedness.
`peautcor`	autocorrelation function of 1-step ahead prediction error.
`pspec`	power spectrum (mtype = 1).

References

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

Examples

```
data(Canadianlynx)
Regressor <- matrix(c( 1, 0, 0, 2, 0, 0, 3, 0,  0, 4, 0,  0, 5, 0,  0, 6, 0, 0, 7,
                       0, 0, 8, 0, 0, 9, 0, 0, 10, 0, 0, 11, 0, 0, 12, 0, 0, 1, 1,
                     0, 2, 2, 0, 1, 2, 0, 3,  3, 0, 1,  1, 1, 2,  2, 2, 3, 3, 3 ), 3,19)
z <- bsubst(Canadianlynx, mtype=2, lag=12, nreg=19, reg=Regressor, cstep=5 )
z$arcoef.bay
```

`Canadianlynx` *Test Data from canadianlynxes*

Description

A test data for "unimar", "unibar", "bsubst" and "exsar".

Usage

```
data(Canadianlynx)
```

Source

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

`canarm` *Canonical Correlation Analysis of Scalar Time Series*

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,...,n) is given by

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

where 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.
`canocoef`	canonical R.
`canocoef2`	R-squared.
`chisquar`	chi-square.
`ndf`	N.D.F.

`dic`	DIC.
`dicmin`	minimum DIC.
`order.dicmin`	order of minimum DIC.
`arcoef`	AR coefficients a(i) (i = 1,...,p).
`macoef`	MA coefficients b(i) (i = 1,...,q).

References

H.Akaike, E.Arahata and T.Ozaki (1975) *Computer Science Monograph, No.5, Timsac74, A Time Series Analysis and Control Program Package (1).* The Institute of Statistical Mathematics.

Examples

```
# "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)
z <- canarm(y, max.order=30)
z$arcoef
z$macoef
```

`canoca` *Canonical Correlation Analysis of Vector Time Series*

Description

Analyze canonical correlation of an 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`	structual characteristic vector H of the canonical Markovian representation.
`matG`	the estimate of the input matrix G.
`vectF`	F matrix in vector form.

References

H.Akaike, E.Arahata and T.Ozaki (1975) *Computer Science Monograph, No.5, Timsac74, A Time Series Analysis and Control Program Package (1).* The Institute of Statistical Mathematics.

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),3,3,byrow=TRUE)
ar[,,2] <- matrix(c(0,  -0.3,  0.5,
                    0.7, -0.4,  1,
                    0,   -0.5,  0.3),3,3,byrow=TRUE)
x <- matrix(rnorm(1000*3),1000,3)
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,...,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

H.Akaike, E.Arahata and T.Ozaki (1975) *Computer Science Monograph, No.5, Timsac74, A Time Series Analysis and Control Program Package (1).* The Institute of Statistical Mathematics.

Examples

```
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, frequency=12, seasonal.order=1,
log=FALSE, trade=FALSE, diff=1, year=1980, month=1, miss=1, omax=99999.9,
plot=TRUE)
```

Arguments

`y`	a univariate time series.
`trend.order`	trend order (0, 1, 2 or 3).
`ar.order`	AR order (less than 11, try 2 first).
`frequency`	number of seasons in one period.
`seasonal.order`	seasonal order (0, 1 or 2).
`log`	log transformation of data (if log = TRUE).
`trade`	trading day adjustment (if trade = TRUE).
`diff`	numerical differencing (1 sided or 2 sided).
`year`	the first year of the data.
`month`	the first month of the data.
`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 trade 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 (m1:trend.order)

$T(t) = T(t-1) + V1(t) : m1 = 1$

$T(t) = 2T(t-1) - T(t-2) + V1(t) : m1 = 2$

$T(t) = 3T(t-1) - 3T(t-2) + T(t-2) + V1(t) : m1 = 3$

AR component (m2:ar.order)

$AR(t) = a(1)AR(t-1) + ... + a(m2)AR(t-m2) + V2(t)$

Seasonal component (k:seasonal.order, f:=frequency)

$S(t) = -S(t-1) - ... - S(t-f+1) + V3(t) : k = 1$

$S(t) = -2S(t-1) - ... - f * S(t-f+1) - ... - S(t-2f+2) + V3(t) : k = 2$

Trading day effect

$TD(t) = b(1)TRADE(t,1) + ... + b(7)TRADE(t,7)$

where $TRADE(t,i)$ is the number of i-th days of the week in t-th data and $b(1) + ... + b(7) = 0$.

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 tau2(1).
`tau2`	system noise variances tau2(2).
`tau3`	system noise variances tau2(3).
`arcoef`	vector of AR coefficients.
`tdf`	trading day factor TDF(i) (i=1,7).

References

G.Kitagawa (1981) *A Nonstationary Time Series Model and Its Fitting by a Recursive Filter* Journal of Time Series Analysis, Vol.2, 103-116.

W.Gersch and G.Kitagawa (1983) *The prediction of time series with Trends and Seasonalities* Journal of Business and Economic Statistics, Vol.1, 253-264.

G.Kitagawa (1984) *A smoothness priors-state space modeling of Time Series with Trend and Seasonality* Journal of American Statistical Association, VOL.79, NO.386, 378-389.

Examples

```
data(Blsallfood)
z <- decomp(y=Blsallfood, trade=TRUE, year=1973)
z$aic
z$lkhd
z$sigma2
z$tau1
z$tau2
z$tau3
```

`exsar` *Exact Maximum Likelihood Method of Scalar AR Model Fitting*

Description

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

Usage

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

Arguments

`y`	a univariate time sries.
`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.
`tmp.file`	a character string naming a file written intermediate results of minimization by DAVIDON-FLETCHER-POWELL procedure. If NULL (default) output no file.

Details

The AR model is given by

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

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.
`arcoef.maice`	MAICE AR coefficients.
`v.mle`	maximum likelihood estimates of innovation variance.
`arcoef.mle`	maximum likelihood estimates of AR coefficients.

References

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

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. isw = 1 : autocorrelation of X (one-channel) isw = 2 : autocorrelations of X and Y (two-channel) isw = 4 : auto- and cross- correlations of X and Y (two-channel)
`plot`	logical. If TRUE (default) crosscorrelations are plotted.
`lag_axis`	logical. If TRUE (default) with plot=TRUE, x_axis is drawn.

Value

`acov`	autocovariance.
`ccov12`	crosscovariance.
`ccov21`	crosscovariance.
`acor`	autocorrelation.
`ccor12`	crosscorrelation.
`ccor21`	crosscorrelation.
`mean`	mean.

References

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

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(0,1000,2)
x[,1] <- xorg[1:1000]
x[,2] <- xorg[4:1003]+0.5*rnorm(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

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

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

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

Examples

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

`fpec` *AR model Fitting for Control*

Description

Perform AR model fitting for control.

Usage

```
fpec(y, max.order=NULL, ncon=NULL, nman=0, inw=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.
`ncon`	number of controlled variables. Default is d, where d is the dimension of the time series y.
`nman`	number of maninpulated variables.
`inw`	indicator; inw[i] (i=1,...,ncon) indicate the controlled variables and inw[i+ncon] (i=1,...,nman) indicate the manipulate variables.

Value

`cov`	covariance matrix rearrangement by inw.
`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

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

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),3,3,byrow=TRUE)
ar[,,2] <- matrix(c(0,  -0.3,  0.5,
                    0.7, -0.4,  1,
                    0,   -0.5,  0.3),3,3,byrow=TRUE)
x <- matrix(rnorm(200*3),200,3)
y <- mfilter(x,ar,"recursive")
fpec(y, max.order=10, ncon=3, nman=0)
```

`LaborData` *Labor force Data*

Description

Labor force U.S. unemploved 16 yeaes or over (1972-1978) data.

Usage

```
data(LaborData)
```

Source

H.Akaike, T.Ozaki, M.Ishiguro, Y.Ogata, G.Kitagawa, Y-H.Tamura, E.Arahata, K.Katsura and Y.Tamura (1985) *Computer Science Monograph, No.22, Timsac84 Part 1.* The Institute of Statistical Mathematics.

`locarData` *Non-stationary Test Data*

Description

A non-stationary test data for "mlocar" and "blocar".

Usage

```
data(locarData)
```

Source

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

`markov` *Maximum Likelihood Computation of Markovian Model*

Description

Compute maximum likelihood estimates of Markovian model.

Usage

```
markov(y, tmp.file=NULL)
```

Arguments

`y`	a multivariate time series.
`tmp.file`	temporary file name. If NULL (default) output no file.

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.
`impuls`	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`	inovation variance.
`aic`	AIC.

References

H.Akaike, E.Arahata and T.Ozaki (1975) *Computer Science Monograph, No.5, Timsac74, A Time Series Analysis and Control Program Package (1).* The Institute of Statistical Mathematics.

Examples

```
x <- matrix(rnorm(1000*2),1000,2)
ma <- array(0,dim=c(2,2,2))
ma[,,1] <- matrix(c( -1.0,  0.0,
                      0.0, -1.0), 2,2,byrow=TRUE)
ma[,,2] <- matrix(c( -0.2,  0.0,
                     -0.1, -0.3), 2,2,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), 2,2,byrow=TRUE)
ar[,,2] <- matrix(c( -0.5, -0.2,
                     -0.2, -0.5), 2,2,byrow=TRUE)
ar[,,3] <- matrix(c( -0.3, -0.05,
                     -0.1, -0.30), 2,2,byrow=TRUE)
z <- mfilter(y,ar,"recursive")
markov(z)
```

`mfilter` *Linear Filtering on a Multivariate Time Series*

Description

Applies linear filtering to a multivariate time series.

Usage

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

Arguments

`x`	a multivariate (m-dimensinal,n length) time series x[n,m].
`filter`	a 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 foe 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 malutivariate 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

$y[i,]' = x[,i]' + f[,,1] * y[i-1,]' + ... + f[,,p] * y[i-p,]'$,

No check is made to see if recursive filter is invertible: the output maydiverge if it is not.

The convolution filter is

$y[i,]' = f[,,1] * x[i,]' + ... + f[,,p] * x[i-p+1,]'$

Value

`mfilter` returns a time series object.

Note

'convolve(, type="folter")' uses the FFT for computations and so *may* be faster foe 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),3,3,byrow=TRUE)
  ar[,,2] <- matrix(c(0,  -0.3,  0.5,
                      0.7, -0.4,  1,
                      0,   -0.5,  0.3),3,3,byrow=TRUE)
  x <- matrix(rnorm(100*3),100,3)
  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),1000,2)
  ma <- array(0,dim=c(2,2,2))
  ma[,,1] <- matrix(c( -1.0,  0.0,
                        0.0, -1.0), 2,2,byrow=TRUE)
  ma[,,2] <- matrix(c( -0.2,  0.0,
                       -0.1, -0.3), 2,2,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), 2,2,byrow=TRUE)
  ar[,,2] <- matrix(c( -0.5, -0.2,
                       -0.2, -0.5), 2,2,byrow=TRUE)
  ar[,,3] <- matrix(c( -0.3, -0.05,
                       -0.1, -0.30), 2,2,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

`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 the basic local span.
`const`	integer. 0 denotes constant vector is not included as a regressor and 1 denotes constant vector is included as the first regressor.
`plot`	logical. If TRUE (default) spectrums pspec are plotted.

Details

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

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

where `ni (i=1,...,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

`mean`	mean.
`var`	variance.
`ns`	the number of local spans.
`order`	order of the current model.
`arcoef`	AR coefficients of current 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.
`pspec`	power spectrum.
`npre`	data length of the preceding stationary block.
`nnew`	data length of the new block.
`order.mov`	order of the moving model.
`v.mov`	innovation variance of the moving model.
`aic.mov`	AIC of the moving model.
`order.const`	order of the constant model.
`v.const`	innovation variance of the constant model.
`aic.const`	AIC of the constant model.

References

G.Kitagawa and H.Akaike (1978) A Procedure for The Modeling of Non-Staionary Time Series. Ann. Inst. Statist. Math., 30, B, 351–363.

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

Examples

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

`mlomar` *Minimum AIC Method of Loccally 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

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

Arguments

`y`	a multivariate 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.
`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 devided into k locally stationary spans,

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

where ni (i=1,...,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. 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

G.Kitagawa and H.Akaike (1978) A Procedure for The Modeling of Non-Stationary Time Series. Ann. Inst. Statist. Math., 30, B, 351–363.

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

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. Default is 2*sqrt(n), where n is the length of the time series y.
`plot`	logical. If TRUE daic is plotted.

Details

The statistic AIC is defined by

$AIC = nlog(det(v)) + 2k$,

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

$W(n) = const * C(m)/(m + 1)$,

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,...,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

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

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

$ek = (D(1)^2 + ... + D(lag)^2)id + id(id + 1)/2$

where id denotes dimension of the process.

Value

`mean`	mean.
`var`	variance.
`v`	innovation variance.
`aic`	AIC(m), (m = 0,...,max.order).
`aicmin`	minimum AIC.
`daic`	AIC(m)-aicmin (m = 0,...,max.order).
`order.maice`	order of minimum AIC.
`v.maice`	innovation variance attained at m=order.maice.
`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

H.Akaike (1978) A Bayesian Extension of The Minimum AIC Procedure of Autoregressive Model Fitting. Research Memo. NO.126, The Institute of Statistical Mathematics.

G.Kiagawa and H.Akaike (1978) A Procedure for The Modeling of Non-stationary Time Series. Ann. Inst. Statist. Math., 30, B, 351–363.

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

Examples

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

`mulcor` *Multiple Correlation*

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

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

Examples

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

# Example 2
xorg <- rnorm(1003)
x <- matrix(0,1000,2)
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

```
mulfrf(y, lag=NULL, niv, iovar=c(1:(niv+1)))
```

Arguments

`y`	a multivariate time series.
`lag`	maximum lag. Default is 2*sqrt(n), where n is the length of the time series y.
`niv`	number of input variables.
`iovar`	input and output variables. iovar(i)-th variables (1<=i<=niv) are input variables and iovar(niv+1)-th variable is output variable.

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

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

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),3,3,byrow=TRUE)
ar[,,2] <- matrix(c(0,  -0.3,  0.5,
                    0.7, -0.4,  1,
                    0,   -0.5,  0.3),3,3,byrow=TRUE)
```

```
x <- matrix(rnorm(200*3),200,3)
y <- mfilter(x,ar,"recursive")
mulfrf(y, lag=20, niv=2)
```

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

Description

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

Usage

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

Arguments

`y`	a multivariate 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.
`plot`	logical. If TRUE daic[[1]],...,daic[[d]] are plotted, where d is the dimention of the multivariate time series.
`tmp.file`	a character string naming a file written intermediate results of AIC minimazation. If NULL (default) output no file.

Details

Multivariate autoregressive model is defined by

$y(t) = A(1)y(t-1) + A(2)y(t-2) + ... + A(p)y(t-p) + u(t),$

where p is order of the model and u(t) is Gaussian white noise with mean 0 and variance matrix matv.

AIC is defined by

$AIC = nlog(det(v)) + 2k$

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(m) (m = 0,...,max.order).

`aicmin`	minimum AIC.
`daic`	AIC(m)-aicmin (m = 0,...,max.order).
`order.maice`	order of minimum AIC.
`v.maice`	innovation variance attained at m = order.maice.
`np`	number of parameters.
`jnd`	specification of i-th regressor.
`subregcoef`	subset regression coefficients.
`rvar`	residual variance.
`aicf`	final estimate of AIC (=nlog(rvar)+2np).
`respns`	instantaneous response.
`matv`	innovation variance matrix.
`morder`	order of the MAICE model.
`arcoef`	AR coefficients. arcoef[i,j,k] shows the value of i-th row, j-th column, k-th order.
`aicsum`	the sum of aicf.

References

G.Kitagawa and H.Akaike (1978) A Procedure for The Modeling of Non-stationary Time Series. Ann. Inst. Statist. Math., 30, B, 351–363.

H.Akaike, G.Kiragawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

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),3,3,byrow=TRUE)
ar[,,2] <- matrix(c(0,  -0.3,  0.5,
                    0.7, -0.4,  1,
                    0,   -0.5,  0.3),3,3,byrow=TRUE)
x <- matrix(rnorm(200*3),200,3)
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, ncon=NULL, nman=0, h, inw=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.
`ncon`	number of controlled variables. Default is d, where d is the dimension of the time series y.
`nman`	number of maninpulated variables.
`h`	specify frequencies i/2h (i=0,...,h).
`inw`	indicator; inw[i] (i=1,...,ncon) indicate the controlled variables and inw[i+ncon] (i=1,...,nman) indicate the manipulate variables.

Value

`nperr`	a normalized prediction error covaiance matrix.
`diffr`	differential relative power contribution.
`integr`	integrated relative power contribution.

References

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

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

`mulrsp` *Multiple Rational Spectrum*

Description

Compute rational spectrum for d-dimensional ARMA process.

Usage

```
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-axisis 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`.

Value

`rspec`	rational spectrum.
`scoh`	simple coherence.

References

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

Examples

```
# Example 1 for the normal distribution
xorg <- rnorm(1003)
x <- matrix(0,1000,2)
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),3,3,byrow=TRUE)
ar[,,2] <- matrix(c(0,  -0.3,  0.5,
                    0.7, -0.4,  1,
                    0,   -0.5,  0.3),3,3,byrow=TRUE)
x <- matrix(rnorm(200*3),200,3)
y <- mfilter(x,ar,"recursive")
z <- fpec(y, max.order=10, ncon=3, nman=0)
mulrsp(h=20, d=3, cov=z$perr, ar=z$arcoef)   # d=ncon+nman
```

`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. (y[n,d])
`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 values 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: Pahse spectrums.
`plot.scale`	logical. IF TRUE the common range of the y-axisis is used.

Details

Hanning Window : a1(0)=0.5, a1(1)=a1(-1)=0.25, a1(2)=a1(-2)=0

Akaike Window : a2(0)=0.625, a2(1)=a2(-1)=0.25, a2(2)=a2(-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

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

Examples

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

`MYE1F` *An earthquake wave data*

Description

An earthquake wave data.

Usage

```
data(MYE1F)
```

Source

G.Kitagawa (1993) *Time series analysis programing* The Iwanami Computer Science Senes.

`ngsmth` *Non-Gaussian Smoothing*

Description

Trend estimation by Non-Gaussian smoothing.

Usage

```
ngsmth(y,noisev=2,tau2,bv=1.0,noisew=1,sig2,bw=1.0,initd=1,k=200,plot=TRUE)
```

Arguments

`y`	a univariate time series.
`noisev`	type of system noise density. (1:Gaussian (normal), 2:Pearson family, 3:two-sides exponential)
`tau2`	variance of dispersion of system noise.
`bv`	shape parameter of system noise (for noisev=2).
`noisew`	type of observation noise density. (1:Gaussian (normal), 2:Pearson family, 3:two-sided exponential, 4:double exponential)
`sig2`	variance of dispersion of observation noise.
`bw`	shape parameter of observation noise (for noisew=2).
`initd`	type of density function. (0:two-sided exponential, 1:Gaussian (normal), 2:uniform)
`k`	number of intervals
`plot`	logical. If TRUE (default) trend and smoothed density are plotted.

Details

Consider a one dimensional state space model

$x(n) = x(n-1) + v(n),$

$y(n) = x(n) + w(n),$

where the observation noise $w(n)$ is assumed to be Gaussian distributed and the system noise $v(n)$ is assumed to be distributed as the Pearson system

$q(v(n)) = c/(\tau^2 + v(n)^2)^b$

with $1/2 < b < \infty$ and $c = \tau^{2b-1}\Gamma(b)/\Gamma(1/2)\Gamma(b-1/2)$.

This broad family of distributions includes the Cauchy distribution (b=1).

Value

`trend`	trend.
`smt`	smoothed density.
`lkhood`	log-likelihood.

References

Kitagawa, G., (1993) *Time series analysis programing (in Japanese).* The Iwanami Computer Science Senes.

Kitagawa, G. and Gersch, W., (1996) *Smoothness Priors Analysis of Time Series.* Lecture Notes in Statistics, No.116, Springer-Verlag.

Examples

```
# trend model
x <- rep(0,400)
x[101:200] <- 1
x[201:300] <- -1
y <- rnorm(600, mean=0, sd=0.5)
y <- y[101:500]
z1 <- ngsmth(x+y, noisev=2, tau2=0.211e-09, bv=1, noisew=2, sig2=1.042, bw=1 )

z2 <- ngsmth(x+y, noisev=1, tau2=0.14e-01, bv=1, noisew=2, sig2=1.048, bw=1 )


#  an earthquake wave data
data(MYE1F)
n <- length(MYE1F)
m <- n/2
y <- rep(0, n)
for( i in 2:n )  y[i] <- MYE1F[i] - 0.5*MYE1F[i-1]
yy <- rep(0, m)
for( i in 1:m ) yy[i] <- y[i*2]
z <- tvvar(yy, trend.order=2, tau20= 6.6e-06, delta=1.0e-06, plot=FALSE)
z1 <- ngsmth( z$ts, noisev=2, tau2=0.00026, bv=1, noisew=2, sig2=1.644934, bw=1, k=190 )

z2 <- ngsmth( z$ts, noisev=1, tau2=0.04909, bv=1, noisew=2, sig2=1.644934, bw=1, k=190 )
```

`nonst` *Non-stationary Power Spectrum Analysis*

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

$y(t) = A(1)y(t-1) + A(2)y(t-2) + ... + A(p)y(t-p) + u(t),$

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

AIC is defined by

$AIC = nlog(det(sd)) + 2k$

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 time series `"y"`).
`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

H.Akaike, E.Arahata and T.Ozaki (1976) *Computer Science Monograph, No.6, Timsac74 A Time Series Analysis and Control Program Package (2)*. The Institute of Statistical Mathematics.

Examples

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

`nonstData` *Non-stationary Test Data*

Description

A non-stationary test data for "nonst".

Usage

```
data(nonstData)
```

Source

H.Akaike, E.Arahata and T.Ozaki (1976) *Computer Science Monograph, No.6, Timsac74 A Time Series Analysis and Control Program Package (2).* The Institute of Statistical Mathematics.

`optdes` *Optimal Controller Design*

Description

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

Usage

```
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 (ir, ir) matrix Q, where ir is the number of controlled variables. A quadratic criterion is defined by Q and R.
`r`	positive definite (il, il) matrix R, where il is th number of maninpulated variables.

Value

`perr`	prediction error covariance matrix.
`trans`	first ir columns of transition matrix, where ir is the number of controlled variables.
`gamma`	gamma matrix.
`gain`	gain matrix.

References

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

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),3,3,byrow=TRUE)
  ar[,,2] <- matrix(c(0,  -0.3,  0.5,
                      0.7, -0.4,  1,
                      0,   -0.5,  0.3),3,3,byrow=TRUE)
  x <- matrix(rnorm(200*3),200,3)
  y <- mfilter(x,ar,"recursive")
  q <- matrix(c(0.16,0,0,0.09), 2, 2)
  r <- matrix(0.001, 1, 1)
  optdes(y,, ns=20, q, r)
```

`optsim` *Optimal Control Simulation*

Description

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

Usage

```
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 ir columns of transition matrix, where ir 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

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

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),3,3,byrow=TRUE)
  ar[,,2] <- matrix(c(0,  -0.3,  0.5,
                      0.7, -0.4,  1,
                      0,   -0.5,  0.3),3,3,byrow=TRUE)
  x <- matrix(rnorm(200*3),200,3)
  y <- mfilter(x,ar,"recursive")
  q <- matrix(c(0.16,0,0,0.09), 2, 2)
  r <- matrix(0.001, 1, 1)
  optsim(y, max.order=10, ns=20, q, r, 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

```
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,...,nd, j=1,...,ni) is defined by

$z(i,j) = y(ni(i-1)+j)$,

$z(i,j) = c(j) + A(1,j,0)z(i,1) + ... + A(j-1,j,0)z(i,j-1) + A(1,j,1)z(i-1,1) + ... + A(ni,j,1)z(i-1,ni) + ... + u(i,j)$,

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

$AIC = nlog(det(v)) + 2k$,

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.
`ord`	specification of i-th regressor (i=1,...,ni).
`regcoef`	regression coefficients.
`rvar`	residual variances.
`np`	number of parameters.

`aic`	AIC.
`v`	innovation variance matrix.
`arcoef`	AR coefficient matrices. arcoef[i,,k] shows i-th regressand of k-th period formar.
`const`	constant vector.
`morder`	order of the MAICE model.

References

M.Pagano (1978) On Periodic and Multiple Autoregressions. Ann. Statist., 6, 1310–1317.

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

Examples

```
data(Airpolution)
z <- perars(Airpolution, ni=6, lag=2, ksw=1)
z$regcoef
z$v
```

`Powerplant` *Power Plant Data*

Description

A Power plant test data; 1-ch command, 2-ch temperature, 3-ch fuel.

Usage

```
data(Powerplant)
```

Source

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

`prdctr` *Prediction Program*

Description

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

Usage

```
prdctr(y, r, s, h, arcoef, macoef=NULL, impuls=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.
`impuls`	impulse response matrices.
`v`	inovation 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) $(r <= i <= s + h)$.
`ys`	predct(i) - y(i) $(r <= i <= n)$.
`pstd`	predct(i) + (standard deviation) $(s <= i <= s + h)$.
`p2std`	predct(i) + 2*(standard deviation) $(s <= i <= s + h)$.
`p3std`	predct(i) + 3*(standard deviation) $(s <= i <= s + h)$.
`mstd`	predct(i) - (standard deviation) $(s <= i <= s + h)$.
`m2std`	predct(i) - 2*(standard deviation) $(s <= i <= s + h)$.
`m3std`	predct(i) - 3*(standard deviation) $(s <= i <= s + h)$.

References

H.Akaike, E.Arahata and T.Ozaki (1975) *Computer Science Monograph, No.6, Timsac74, A Time Series Analysis and Control Program Package (2).* The Institute of Statistical Mathematics.

Examples

```
# "arima.sim" is a function in "stats".
# Note that the sign of MA coefficient is opposite from that in "timsac".
yy <- arima.sim(list(order=c(2,0,1), ar = c(0.64,-0.8), ma=c(-0.5)), n=350)
y1 <- yy[51:300]
z <- autoarmafit(y1)
ar <- z$model[[1]]$arcoef
ma <- z$model[[1]]$macoef
v <- z$model[[1]]$v
y2 <- yy[301:350]
prdctr(y2, r=30, s=50, h=10, arcoef=ar, macoef=ma, v=v)
```

`raspec` *Rational Spectrum*

Description

Compute power spectrum of ARMA process.

Usage

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

Arguments

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

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 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

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

Examples

```
# 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

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

Arguments

`y`	a multivariate time series.
`lag`	maximum lag. Default is 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`	coherncy.
`freqr`	frequency response function : real part.

`freqi`	frequency response function : imaginary part.
`errstat`	relative error statistics.
`phase`	phase.

References

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

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),3,3,byrow=TRUE)
ar[,,2] <- matrix(c(0,  -0.3,  0.5,
                    0.7, -0.4,  1,
                    0,   -0.5,  0.3),3,3,byrow=TRUE)
x <- matrix(rnorm(200*3),200,3)
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

```
simcon(span, len, r, arcoef, impuls, 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.
`impuls`	impulse responce 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);

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

where A(i) (i=1,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`	(p*d,r) sub matrices 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`	(p*d,d-r) sub matrices of impulse response matrices.

References

H.Akaike, E.Arahata and T.Ozaki (1975) *Computer Science Monograph, No.6, Timsac74, A Time Series Analysis and Control Program Package (2).* The Institute of Statistical Mathematics.

Examples

```
x <- matrix(rnorm(1000*2),1000,2)
ma <- array(0,dim=c(2,2,2))
ma[,,1] <- matrix(c( -1.0,  0.0,
                      0.0, -1.0), 2,2,byrow=TRUE)
ma[,,2] <- matrix(c( -0.2,  0.0,
                     -0.1, -0.3), 2,2,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), 2,2,byrow=TRUE)
ar[,,2] <- matrix(c( -0.5, -0.2,
                     -0.2, -0.5), 2,2,byrow=TRUE)
ar[,,3] <- matrix(c( -0.3, -0.05,
                     -0.1, -0.30), 2,2,byrow=TRUE)
y <- mfilter(y,ar,"recursive")
z <- markov(y)
weight <-  matrix(c(0.0002,  0.0,
                    0.0,    2.9 ), 2,2,byrow=TRUE)
simcon(span=50, len=700, r=1, z$arcoef, z$impuls, z$v, weight)
```

`thirmo` *Third Order Monents*

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

H.Akaike, E.Arahata and T.Ozaki (1975) *Computer Science Monograph, No.6, Timsac74, A Time Series Analysis and Control Program Package (2)*. The Institute of Statistical Mathematics.

Examples

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

`tsmooth` *Kalman Filter*

Description

State estimation of user-defined state space model by Kalman filter.

Usage

```
tsmooth(y,f,g,h,q,r,x0=NULL,v0=NULL,filter.end=NULL,predict.end=NULL,outmin=-10.0e+30,outmax=10.0e
```

Arguments

`y`	l-dimensional time series y(n).
`f`	$m*m$ state transition matrix F(n), where m is the dimension of the state vector x(n).
`g`	$m*k$ matrix G(n), where k is the demension of the system noise.
`h`	$l*m$ matrix H(n).
`q`	system noise variance Q(n).
`r`	observational noise variance R(n).
`x0`	initial state vector X(0|0).
`v0`	initial state covariance matrix V(0|0).
`filter.end`	end point of filtering.
`predict.end`	end point of prediction.
`outmin`	lower limits of observations.
`outmax`	upper limits of observations.
`missed`	start position of missed intervals.
`np`	number of missed observations.
`plot`	logical. If TRUE estimated smoothed state is plotted.

Details

The linear Gaussian state space model is

$x(n) = F(n)x(n-1) + G(n)v(n),$

$y(n) = H(n)x(n) + w(n),$

where $y(n)$ is an l-dimensional time series, $x(n)$ is m-dimensional state vector. $v(n)$ and $w(n)$ are k- and l-dimensional white noise sequences. $F(n)$, $G(n)$ and $H(n)$ are $m*m$, $m*k$ and $l*m$ matrices, respectively. $R(n)$ and $Q(n)$ are $k*k$ and $l*l$ matrices, respectively. We asusume that $E(v(n), w(n)) = 0$, $v(n)\tilde{}N(0, Q(n))$ and $w(n)\tilde{}N(0, R(n))$. User should give all the matrices of a state space model and its parameters. In current version, F(n), G(n), H(n), Q(n), R(n) should be time invariant.

Value

`mean.smooth`	mean vectors of the smoother.
`cov.smooth`	variance of the smoother.
`esterr`	estimation error.
`lkhood`	log-likelihood.
`aic`	AIC.

References

Kitagawa, G., (1993) *Time series analysis programing (in Japanese).* The Iwanami Computer Science Senes.

Kitagawa, G. and Gersch, W., (1996) *Smoothness Priors Analysis of Time Series.* Lecture Notes in Statistics, No.116, Springer-Verlag.

Examples

```
## AR model (l=1, m=10, k=1)
#  m <- 5
  m <- 10
  k <- 1
  data(Blsallfood)
  z1 <- exsar(Blsallfood, max.order=m)
  var <- z1$var
  tau2 <- z1$v.mle
  arcoef <- z1$arcoef.mle
  f <- matrix( 0.0e0, m, m )
  f[1,] <- arcoef
  for( i in 2:m ) f[i,i-1] <- 1
  g <- c(1, rep(0.0e0, m-1))
  h <- c(1, rep(0.0e0, m-1))
  q <- tau2
  r <- 0.0e0
  x0 <- rep(0.0e0, m)
  v0 <- matrix( 0.0e0, m, m )
  for( i in 1:m ) v0[i,i] <- var
 z <- tsmooth(Blsallfood, f, g, h, q, r, x0, v0, filter.end=156, predict.end=170, missed=c(41,101), np=c(30,20))

# plot mean vector and estimation error
  xss <- z$mean.smooth[1,] + mean(Blsallfood)
  cov <- z$cov.smooth
  c1 <- xss + sqrt(cov[1,])
  c2 <- xss - sqrt(cov[1,])
  err <- z$esterr
  par(mfcol=c(2,1))
  ymax <- as.integer(max(xss,c1,c2)+1)
  ymin <- as.integer(min(xss,c1,c2)-1)
 plot(c1, type=’l’, ylim=c(ymin,ymax), col=2, xlab="Mean vectors of the smoother XSS(1,) +/- standard deviation",
  par(new=TRUE)
  plot(c2, type=’l’, ylim=c(ymin,ymax), col=3, xlab="", ylab="")
  par(new=TRUE)
```

```
    plot(xss, type='l', ylim=c(ymin,ymax), xlab="", ylab="")
    plot(err[,1,1], type='h', xlim=c(1,length(xss)),xlab="estimation error", ylab="")

  ## Trend model (l=3, m=2, k=2)
    l <- 3
  #  n <- 400
    n <- 500
    m <- 2
    k <- 2
    f <- matrix( c(1, 0,   0, 1), nrow=m, ncol=m, byrow=TRUE )
    g <- matrix( c(1, 0,   0, 1), nrow=m, ncol=k, byrow=TRUE )
    h <- matrix( c(0.1, -0.1,   -0.05, 0.05,   0.2, 0.15), nrow=l, ncol=m, byrow=TRUE)
    q <- matrix( c(0.2*0.2, 0,   0, 0.3*0.3), nrow=k, ncol=k, byrow=TRUE)
   r <- matrix( c(0.2*0.2, 0, 0,  0, 0.1*0.1, 0,  0, 0, 0.15*0.15), nrow=l, ncol=l, byrow=TRUE)
    Xn <- matrix( 0, nrow=m, ncol=n )
    x1 <- rnorm(n+100, 0, 0.2)
    x2 <- rnorm(n+100, 0, 0.3)
    x1 <- cumsum(x1)[101:(n+100)]
    x2 <- cumsum(x2)[101:(n+100)]
    Xn[1,] <- x1-mean(x1)
    Xn[2,] <- x2-mean(x2)
    Yn <- matrix( 0, nrow=l, ncol=n )
    Wn <- matrix( 0, nrow=l, ncol=n )
    Wn[1,] <- rnorm(n, 0, 0.2)
    Wn[2,] <- rnorm(n, 0, 0.1)
    Wn[3,] <- rnorm(n, 0, 0.15)
    Yn <- h %*% Xn + Wn
    Yn <- aperm(Yn, c(2,1))
    x0 <- c(Xn[1,1], Xn[2,1])
    v0 <- matrix( c(var(Yn[,1]), 0,   0, var(Yn[,2])), 2, 2, byrow=TRUE)
    npe <- n+20
   z <- tsmooth(Yn, f, g, h, q, r, x0, v0, filter.end=n, predict.end=npe, missed=n/2, np=n/20)

  # plot mean vector and state vector
    xss <- z$mean.smooth
    par(mfcol=c(m,1))
    for( i in 1:m ) {
      ymax <- as.integer(max(xss[i,],Xn[i,])+1)
      ymin <- as.integer(min(xss[i,],Xn[i,])-1)
     plot(Xn[i,], type='l', xlim=c(1,npe), ylim=c(ymin,ymax), xlab=paste(" red : mean.smooth[",i,",] / black : Xn["
      par(new=TRUE)
      plot(xss[i,], type='l', ylim=c(ymin,ymax), xlab="", ylab="", col=2)
    }
```

---

tvar *Time Varying Coefficients AR model*

Description

Estimate time varying coefficients AR model.

Usage

```
tvar(y, ar.order, trend.order=2, span, outlier, tau20=NULL, delta=NULL, plot=TRUE)
```

Arguments

`y`	a univariate time series.
`ar.order`	AR order.
`trend.order`	trend order (=1 or 2).
`span`	local stationary span.
`outlier`	positions of outliers.
`tau20`	initial value for computing variance of the system noise tau2.
`delta`	delta for computing variance of the system noise tau2. If tau20 is NULL or delta is NULL, tau2 is computed automatically.
`plot`	logical. If TRUE (default) parcor is plotted.

Details

The time-varying coefficients AR model is given by

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

where $a(i,t)$ is i-lag AR coefficient at time t and $u(t)$ is a zwro mean white noise.

Value

`tau2`	variance of the system noise.
`sigma2`	variance of the observational noise.
`lkhood`	log-likelihood.
`aic`	AIC.
`arcoef`	time varying AR coefficients.
`parcor`	partial autocorrelation coefficient.
`spec`	time varying spectrum.

References

Kitagawa, G. (1993) *Time series analysis programing (in Japanese).* The Iwanami Computer Science Senes.

Kitagawa, G. and Gersch, W. (1996) *Smoothness Priors Analysis of Time Series.* Lecture Notes in Statistics, No.116, Springer-Verlag.

Kitagawa, G. and Gersch, W. (1985) *A smoothness priors time varying AR coefficient modeling of nonstationary time series.* IEEE trans. on Automatic Controle, AC-30, 48-56.

Examples

```
 data(MYE1F) # an earthquake wave data
 z <- tvvar(MYE1F, trend.order=2, tau20= 6.6e-06, delta=1.0e-06, plot=FALSE)
zz <- tvar(z$normdat, ar.order=4, trend.order=2, span=20, tau20=6.6e-06, delta=1.0e-06, outlier=c(630,1026))
 zz$tau2
 zz$sigma2
 zz$lkhood
 zz$aic
```

`tvvar` *Time Varying Variance*

Description

Estimate time-varying variance.

Usage

```
tvvar(y, trend.order, tau20, delta, plot=TRUE)
```

Arguments

`y`	univariate time series.
`trend.order`	trend order.
`tau20`	initial estimate of tau2.
`delta`	search width.
`plot`	logical. If TRUE (default) normdat, ts, trend and noise are plotted.

Details

A chi-square distribution with digree 2 is given by

$s(m) = y(2m-1)**2 + y(2m)**2$

where $y(n)$ is original scalar time series and $\sigma(2m-1)**2 = \sigma(2m)**2$.

$z(m) = log(s(m)/2)$.

$z(m) = log(\sigma**2) + w(m)$,

where $w(m)$ is a double exponential distribution with density $h(w) = expw - e**w$.

The space state model is given by

$z(m) = t(m) + w(m)$.

Value

`tvvar`	time varying variance.
`normdat`	normalized data.
`ts`	tranceformed time series s(m).
`trend`	trend.
`noise`	residuals.
`tau2`	variance of the system noise tau2.
`sigma2`	variance of the observational noise.
`lkhood`	log-likelihood of the mode.
`aic`	AIC.

References

Kitagawa, G. (1993) *Time series analysis programing (in Japanese)*. The Iwanami Computer Science Senes.

Kitagawa, G. and Gersch, W. (1996) *Smoothness Priors Analysis of Time Series*. Lecture Notes in Statistics, No.116, Springer-Verlag.

Kitagawa, G. and Gersch, W. (1985) *A smoothness priors time varying AR coefficient modeling of nonstationary time series*. IEEE trans. on Automatic Controle, AC-30, 48-56.

Examples

```
data(MYE1F) # an earthquake wave data
z <- tvvar(MYE1F, trend.order=2, tau20= 6.6e-06, delta=1.0e-06)
z$lkhood
z$aic
```

`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, pacoef and pspec are plotted.

Details

The AR model is given by

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

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

$AIC = nlog(det(v)) + 2m$,

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

$W(m) = CONST * C(m)/(m+1)$,

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

$ek = D(1)^2 + ... + D(k)^2 + 1$

where D(j) is defined by $D(j) = W(j) + ... + W(k)$.

m in the definition of AIC is replaced by ek to be define an equivalent AIC for a Bayesian model.

Value

`mean`	mean.
`var`	variance.
`v`	innovation variance.
`aic`	AIC(m) (m = 0,...,ar.order).
`aicmin`	minimum AIC.
`daic`	AIC(m)-aicmin (m = 0,...,ar.order).
`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

H.Akaike (1978) A Bayesian Extension of The Minimum AIC Procedure of Autoregressivemodel Fitting. Research memo. No.126. The Institute of Statistical Mathematics.

G.Kitagawa and H.Akaike (1978) A Procedure for The Modeling of Non-Stationary Time Series. Ann. Inst. Statist. Math., 30, B, 351–363.

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

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, tmp.file=NULL)
```

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.
`tmp.file`	a character string naming a file written intermediate results of AR coefficients computation. If NULL (default) output no file.

Details

The AR model is given by

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

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

AIC is defined by

$AIC = nlog(det(v)) + 2k$

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(m) (m = 0,...,max.order).
`aicmin`	minimum AIC.
`daic`	AIC(m)-aicmin (m = 0,...,max.order).
`order.maice`	order of minimum AIC.
`v.maice`	innovation variance attained at "`order.maice`".
`arcoef`	AR coefficients.

References

G.Kitagawa and H.Akaike (1978) A Procedure For The Modeling of Non-Stationary Time Series. Ann. Inst. Statist. Math.,30, B, 351–363.

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

Examples

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

`wnoise` *White Noise Generator*

Description

Genetate 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

H.Akaike and T.Nakagawa (1988) *Statistical Analysis and Control of Dynamic Systems.* Kluwer Academic publishers.

Examples

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

# Example 2
v <- matrix(c(1,  0,  0,
              0,  2,  0,
              0,  0,  3),3,3,byrow=TRUE)
wnoise(len=20, perr=v )
```

`xsarma` *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

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

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

H.Akaike (1978) Covariance matrix computation of the state variable of a stationary Gaussian process. Research Memo. No.139. The Institute of Statistical Mathematics.

H.Akaike, G.Kitagawa, E.Arahata and F.Tada (1979) *Computer Science Monograph, No.11, Timsac78.* The Institute of Statistical Mathematics.

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)
macoefi <- c(0.0)
z <- xsarma(y, arcoefi, macoefi)
z$arcoef
z$macoef
```

Index