Package ‘TSSS’

April 22, 2019

Version 1.2.4

Title Time Series Analysis with State Space Model

Author The Institute of Statistical Mathematics, based on the program by Genshiro Kitagawa

Maintainer Masami Saga <msaga@mtb.biglobe.ne.jp>

Depends R (>= 3.4.0), datasets, stats

Suggests utils

Imports graphics

Description Functions for statistical analysis, modeling and simulation of time series with state space model, based on the methodology in Kitagawa (1993, ISBN: 4-00-007703-1 and 2005, ISBN: 4-00-005455-4).

License GPL (>= 2)

MailingList Please send bug reports to ismrp@jasp.ism.ac.jp

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-04-22 11:40:03 UTC

R topics documented:

`TSSS-package` *Time Series Analysis with State Space Model*

Description

R functions for statistical analysis, modeling and simulation of time series with state space model.

Details

This package provides functions for statistical analysis, modeling and simulation of time series. These functions are developed based on source code of "FORTRAN 77 Programming for Time Series Analysis".

Now, a revised edition "Introduction to Time Series Analysis (in Japanese)" and "Introduction to Time Series Modeling" are published.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

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

Kitagawa, G. (2005) *Introduction to Time Series Analysis (in Japanese)*. Iwanami Publishing Company.

Kitagawa, G. (1993) *FORTRAN 77 Programming for Time Series Analysis (in Japanese)*. The Iwanami Computer Science Series.

`arfit` *Univariate AR Model Fitting*

Description

Fit a univariate AR model by Yule-Walker method, Least squares (Householder) method or PARCOR method.

Usage

```
arfit(y, lag = NULL, method = 1, plot = TRUE, ...)
```

Arguments

`y`	a univariate time series.
`lag`	highest order of AR model. Default is $2\sqrt{n}$, where n is the length of the time series `y`.
`method`	estimation procedure. 1 : Yule-Walker method 2 : Least squares (Householder) method 3 : PARCOR method (Partial autoregression) 4 : PARCOR method (PARCOR) 5 : PARCOR method (Burg's algorithm)
`plot`	logical. If `TRUE` (default), PARCOR, AIC and power spectrum are plotted.
`...`	further arguments to be passed to `plot.arfit`.

Value

An object of class `"arfit"`, which is a list with the following elements:

`sigma2`	innovation variance.
`maice.order`	order of minimum AIC.
`aic`	AIC.

arcoef	AR coefficients of the best model.
parcor	PARCOR.
spec	power spectrum (in log scale).
tsname	the name of the univariate time series y.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# Sun spot number data
data(Sunspot)
arfit(log10(Sunspot), 20)

# BLSALLFOOD data
data(BLSALLFOOD)
arfit(BLSALLFOOD)
```

---

armafit *Scalar ARMA Model Fitting*

Description

Fit a scalar ARMA model by maximum likelihood method.

Usage

```
armafit(y, ar.order, ar = NULL, ma.order, ma = NULL)
```

Arguments

y	a univariate time series.
ar.order	AR order.
ar	initial AR coefficients. If NULL (default), use default initial values.
ma.order	MA order.
ma	initial MA coefficients. If NULL (default), use default initial values.

Value

sigma2	innovation variance.
llkhood	log-likelihood of the model.
aic	AIC of the model.
arcoef	AR coefficients.
macoef	MA coefficients.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# Sun spot number data
data(Sunspot)
y <- log10(Sunspot)
z1 <- armafit(y, ar.order = 3, ma.order = 3)
z1

nd <- length(y)
armaimp(arcoef = z1$arcoef, macoef = z1$macoef, v = z1$sigma2, n = nd, lag = 20)
```

---

armaimp — *Calculate Characteristics of Scalar ARMA Model*

Description

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

Usage

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

Arguments

`arcoef`	AR coefficients.
`macoef`	MA coefficients.
`v`	innovation 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, parcor and characteristic roots are plotted.
`...`	further arguments to be passed to `plot.arma`.

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.

Characteristic roots of AR / MA operator is a list with the following components:

- re: real part R
- im: imaginary part I
- amp: $\sqrt{R^2 + I^2}$
- atan: $\arctan(I/R)$
- degree

Value

An object of class "arma", which is a list with the following elements:

impuls	impulse response function.
acov	autocovariance function.
parcor	partial autocorrelation function.
spec	power spectrum.
croot.ar	characteristic roots of AR operator. See Details.
croot.ma	characteristic roots of MA operator. See Details.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# AR model : y(n) = a(1)*y(n-1) + a(2)*y(n-2) + v(n)
a <- c(0.9 * sqrt(3), -0.81)
armaimp(arcoef = a, v = 1.0, n = 1000, lag = 20)

# MA model : y(n) = v(n) - b(1)*v(n-1) - b(2)*v(n-2)
b <- c(0.9 * sqrt(2), -0.81)
armaimp(macoef = b, v = 1.0, n = 1000, lag = 20)

# ARMA model :  y(n) = a(1)*y(n-1) + a(2)*y(n-2)
#                     + v(n) - b(1)*v(n-1) - b(2)*v(n-2)
armaimp(arcoef = a, macoef = b, v = 1.0, n = 1000, lag = 20)
```

`BLSALLFOOD` *BLSALLFOOD Data*

Description

The monthly time series of the number of workers engaged in food industries in the United States (January 1967 - December 1979).

Usage

```
data(BLSALLFOOD)
```

Format

A time series of 156 observations.

Source

The data were obtained from the United States Bureau of Labor Statistics (BLS).

`boxcox` *Box-Cox Transformation*

Description

Computes Box-Cox transformation and find an optimal lambda with minimum AIC.

Usage

```
boxcox(y, plot = TRUE, ...)
```

Arguments

`y`	a univariate time series.
`plot`	logical. If `TRUE` (default), original data and transformed data with minimum AIC are plotted.
`...`	further arguments to be passed to `plot.boxcox`.

Value

An object of class "boxcox", which is a list with the following elements:

mean	mean of original data.
var	variance of original data.
aic	AIC of the model with respect to the original data.
llkhood	log-likelihood of the model with respect to the original data.
z	transformed data.
aic.z	AIC of the model with respect to the transformed data.
llkhood.z	log-likelihood of the model with respect to the transformed data.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# Sun spot number data
data(Sunspot)
boxcox(Sunspot)

# Wholesale hardware data
data(WHARD)
boxcox(WHARD)
```

---

crscor — *Cross-Covariance and Cross-Correlation*

Description

Computes cross-covariance and cross-correlation functions of the multivariate time series.

Usage

```
crscor(y, lag = NULL, outmin = NULL, outmax = NULL, plot = 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.
outmin	bound for outliers in low side. A default value is -1.0e+30 for each dimension.
outmax	bound for outliers in high side. A default value is 1.0e+30 for each dimension.
plot	logical. If TRUE (default), cross-correlations are plotted.
...	further arguments to be passed to plot.crscor.

Value

An object of class "crscor", which is a list with the following elements:

cov	cross-covariances.
cor	cross-correlations.
mean	mean.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# Yaw rate, rolling, pitching and rudder angle of a ship
data(HAKUSAN)
y <- as.matrix(HAKUSAN[, 2:4])   # Rolling, Pitching, Rudder
crscor(y, lag = 50)
```

---

fftper — *Compute a Periodogram via FFT*

Description

Compute a periodogram of the univariate time series via FFT.

Usage

```
fftper(y, window = 1, plot = TRUE, ...)
```

Arguments

y	a univariate time series.
window	smoothing window type. (0: box-car, 1: Hanning, 2: Hamming)
plot	logical. If TRUE (default), smoothed periodogram is plotted.
...	further arguments to be passed to plot.spg.

Details

$$\begin{array}{rll} \text{Hanning Window :} & W_0 = 0.5 & W_1 = 0.25 \\ \text{Hamming Window :} & W_0 = 0.54 & W_1 = 0.23 \end{array}$$

Value

An object of class "spg", which is a list with the following elements:

`period`	periodogram (raw spectrum).
`smoothed.period`	smoothed periodogram. If there is not a negative number, logarithm of smoothed periodogram.
`log.scale`	if `TRUE` "smooth the periodogram on log scale.
`tsname`	the name of the univariate time series `y`.

Note

We assume that the length N of the input time series `y` is a power of 2. If N is not a power of 2, calculate using the FFT by appending 0's behind the data `y`.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# Yaw rate, rolling, pitching and rudder angle of a ship
data(HAKUSAN)
YawRate <- HAKUSAN[, 1]
fftper(YawRate, window = 0)
```

`HAKUSAN` *Ship's Navigation Data*

Description

A multivariate time series of a ship's yaw rate, rolling, pitching and rudder angles which were recorded every second while navigating across the Pacific Ocean.

Usage

```
data(HAKUSAN)
```

Format

A data frame with 1000 observations on the following 4 variables.

[, 1]	YawRate	yaw rate
[, 2]	Rolling	rolling
[, 3]	Pitching	pitching
[, 4]	Rudder	rudder angle

Source

The data were offered by Prof. K. Ohtsu of Tokyo University of Marine Science and Technology.

klinfo — *Kullback-Leibler Information*

Description

Computes Kullback-Leibler information.

Usage

```
klinfo(distg = 1, paramg = c(0, 1), distf = 1, paramf, xmax = 10)
```

Arguments

`distg` function for the true density (1 or 2).

1 :	Gaussian (normal) distribution paramg(1): mean paramg(2): variance
2 :	Cauchy distribution paramg(1): μ (location parameter) paramg(2): τ^2 (dispersion parameter)

`paramg` parameter vector of true density.

`distf` function for the model density (1 or 2).

1 :	Gaussian (normal) distribution paramf(1): mean paramf(2): variance
2 :	Cauchy distribution paramf(1): μ (location parameter) paramf(2): τ^2 (dispersion parameter)

`paramf` parameter vector of the model density.

`xmax` upper limit of integration. lower limit xmin = `-xmax`.

Value

`nint` number of function evaluation.

`dx` delta.

`KLI` Kullback-Leibler information, $I(g; f)$.

`gint` integration of $g(y)$ over [`-xmax`, `xmax`].

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# g:Gauss, f:Gauss
klinfo(distg = 1, paramg = c(0, 1), distf = 1, paramf = c(0.1, 1.5), xmax = 8)

# g:Gauss, f:Cauchy
klinfo(distg = 1, paramg = c(0, 1), distf = 2, paramf = c(0, 1), xmax = 8)
```

`lsar` *Decomposition of Time Interval to Stationary Subintervals*

Description

Decompose time series to stationary subintervals and estimate local spectrum.

Usage

```
lsar(y, max.arorder = 20, ns0, plot = TRUE, ...)
```

Arguments

`y`	a univariate time series.
`max.arorder`	highest order of AR model.
`ns0`	basic local span.
`plot`	logical. If `TRUE` (default), local spectra are plotted.
`...`	further arguments to be passed to `plot.lsar`.

Value

An object of class `"lsar"`, which is a list with the following elements:

`model`	1: pooled model is accepted. 2: switched model is accepted.
`ns`	number of observations of local span.
`span`	start points and end points of local spans.
`nf`	number of frequencies.
`ms`	order of switched model.
`sds`	innovation variance of switched model.
`aics`	AIC of switched model.
`mp`	order of pooled model.
`sdp`	innovation variance of pooled model.
`aics`	AIC of pooled model.
`spec`	local spectrum.
`tsname`	the name of the univariate time series `y`.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# seismic data
data(MYE1F)
lsar(MYE1F, max.arorder = 10, ns0 = 100)
```

---

lsar.chgpt — *Estimation of the Change Point*

Description

Precisely estimate a change point of subinterval for locally stationary AR model.

Usage

```
lsar.chgpt(y, max.arorder = 20, subinterval, candidate, plot = TRUE, ...)
```

Arguments

`y`	a univariate time series.
`max.arorder`	highest order of AR model.
`subinterval`	a vector of the form `c(n0, ne)` which gives a start and end point of time interval used for model fitting.
`candidate`	a vector of the form `c(n1, n2)` which gives minimum and maximum for change point. `n0+2k` < `n1` < `n2+k` < `ne`, (`k` is `max.arorder`)
`plot`	logical. If `TRUE` (default), `y[n0:ne]` and '`aic`' are plotted.
`...`	further arguments to be passed to `plot.chgpt`.

Value

An object of class `"chgpt"`, which is a list with the following elements:

`aic`	AICs of the AR model fitted on `[n1, n2]`.
`aicmin`	minimum AIC.
`change.point`	a change point.
`subint`	original sub-interval data and information.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# seismic data
data(MYE1F)
lsar.chgpt(MYE1F, max.arorder = 10, subinterval = c(200, 1000),
           candidate = c(400, 800))

lsar.chgpt(MYE1F, max.arorder = 10, subinterval = c(600, 1400),
           candidate = c(800, 1200))
```

---

lsqr *The Least Squares Method via Householder Transformation*

Description

Compute Regression coefficients of the model with minimum AIC.

Usage

```
lsqr(y, lag = 10, plot = TRUE, ...)
```

Arguments

`y`	a univariate time series.
`lag`	number of sine and cosine terms.
`plot`	logical. If `TRUE` (default), original data and fitted trigonometric polynomial are plotted.
`...`	further arguments to be passed to `plot.lsqr`.

Value

An object of class `"lsqr"`, which is a list with the following elements:

`aic`	AIC's of the model with order $0, \ldots, k (= 2\texttt{lag}+1)$.
`sigma2`	residual variance of the model with order $0, \ldots, k$.
`maice.order`	order of minimum AIC.
`regress`	regression coefficients of the model.
`tripoly`	trigonometric polynomial.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# The daily maximum temperatures for Tokyo
data(Temperature)
lsqr(Temperature)
```

marfit — *Yule-Walker Method of Fitting Multivariate AR Model Fitting*

Description

Fit a multivariate AR model by Yule-Walker method.

Usage

```
marfit(y, lag = NULL)
```

Arguments

`y`	a multivariate time series.
`lag`	highest order of fitted AR models. Default is $2\sqrt{n}$, where n is the length of the time series `y`.

Value

An object of class `"maryule"`, which is a list with the following elements:

`maice.order`	order of minimum AIC.
`aic`	AIC's of the AR model with order $0, \ldots,$`lag`.
`v`	innovation covariance matrix of AIC best model.
`arcoef`	AR coefficient of the AIC best model.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# Yaw rate, rolling, pitching and rudder angle of a ship
data(HAKUSAN)
yy <- as.matrix(HAKUSAN[, c(1,2,4)])   # Yaw rate, Pitching, Rudder angle
nc <- dim(yy)[1]
n <- seq(1, nc, by = 2)
y <- yy[n, ]
marfit(y, 20)
```

marlsq — *Least Squares Method for Multivariate AR Model*

Description

Fit a multivariate AR model by least squares method.

Usage

```
marlsq(y, lag = NULL)
```

Arguments

y	a multivariate time series.
lag	highest AR order. Default is $2\sqrt{n}$, where n is the length of the time series y.

Value

An object of class "marlsq", which is a list with the following elements:

maice.order	order of the MAICE model.
aic	total AIC of the model.
v	innovation covariance matrix.
arcoef	AR coefficient matrices.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# Yaw rate, rolling, pitching and rudder angle of a ship
data(HAKUSAN)
y <- as.matrix(HAKUSAN[, c(1,2,4)])   # Yaw rate, Rolling, Rudder angle
z <- marlsq(y)
z

marspc(z$arcoef, v = z$v)
```

marspc *Cross Spectra and Power Contribution*

Description

Compute cross spectra and power contribution.

Usage

```
marspc(arcoef, v, plot = TRUE, ...)
```

Arguments

`arcoef`	AR coefficient matrices.
`v`	innovation variance matrix.
`plot`	logical. If `TRUE` (default), cross spectra and power contribution are plotted.
`...`	further arguments to be passed to `plot.marspc`.

Value

An object of class `"marspc"`, which is a list with the following elements:

`spec`	cross spectra.
`amp`	amplitude spectra.
`phase`	Phase spectra.
`coh`	simple coherency.
`power`	power contribution.
`rpower`	relative power contribution.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# Yaw rate, rolling, pitching and rudder angle of a ship
data(HAKUSAN)
yy <- as.matrix(HAKUSAN[, c(1,2,4)])
nc <- dim(yy)[1]
n <- seq(1, nc, by = 2)
y <- yy[n, ]
z <- marfit(y, lag = 20)

marspc(z$arcoef, v = z$v)
```

MYE1F *Seismic Data*

Description

The time series of East-West components of seismic waves, recorded every 0.02 seconds.

Usage

```
data(MYE1F)
```

Format

A time series of 2600 observations.

Source

Takanami, T. (1991), "ISM data 43-3-01: Seismograms of foreshocks of 1982 Urakawa-Oki earthquake", Ann. Inst. Statist. Math., 43, 605.

ngsim *Simulation by Non-Gaussian State Space Model*

Description

Simulation by non-Gaussian state space model.

Usage

```
ngsim(n = 200, trend = NULL, seasonal.order = 0, seasonal = NULL, arcoef = NULL,
      ar = NULL, noisew = 1, wminmax = c(-1, 1), paramw = NULL, noisev = 1,
      vminmax = c(-1, 1), paramv = NULL, seed = NULL, plot = TRUE, ...)
```

Arguments

`n`	the number of simulated data.
`trend`	initial values of trend component of length at most 2.
`seasonal.order`	seasonal order. (0 or 1)
`seasonal`	if `seasonal.order` > 0, initial values of seasonal component of length $p - 1$, where p is the number of season in one period.
`arcoef`	AR coefficients.
`ar`	initial values of AR component.
`noisew`	type of the observational noise.

-1 :	Cauchy random number (without an inverse function)
-2 :	exponential distribution (without an inverse function)
-3 :	double exponential distribution (without an inverse function)
0 :	double exponential distribution (+ Euler's constant)
1 :	normal distribution,
2 :	Pearson distribution,
3 :	double exponential distribution

`wminmax` lower and upper bound of observational noise.

`paramw` parameter of the observational noise density.

noisew = 1 :	variance
noisew = 2 :	dispersion parameter (tau square), shape parameter

`noisev` type of the system noise.

-1 :	Cauchy random number (without an inverse function)
-2 :	exponential distribution (without an inverse function)
-3 :	double exponential distribution (without an inverse function)
0 :	double exponential distribution (+ Euler's constant)
1 :	normal distribution
2 :	Pearson distribution
3 :	double exponential distribution

`vminmax` lower and upper bound of system noise.

`paramv` parameter of the system noise density.

noisev = 1 :	variance
noisev = 2 :	dispersion parameter (tau square), shape parameter

`seed` arbitrary positive integer to generate a sequence of uniform random numbers. The default seed is based on the current time.

`plot` logical. If `TRUE` (default), simulated data are plotted.

`...` further arguments to be passed to `plot.simulate`.

Value

An object of class `"simulate"`, giving simulated data of non-Gaussian state space model.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
ar1 <- ngsim(n = 400, arcoef = 0.95, noisew = 1, paramw = 1, noisev = 1,
             paramv = 1, seed = 555)
```

```
plot(ar1, use = c(201, 400))

ar2 <- ngsim(n = 400, arcoef = c(1.3, -0.8), noisew = 1, paramw = 1, noisev = 1,
             paramv = 1, seed = 555)
plot(ar2, use = c(201, 400))
```

---

ngsmth *Non-Gaussian Smoothing*

Description

Trend estimation by non-Gaussian smoothing.

Usage

```
ngsmth(y, noisev = 2, tau2, bv = 1.0, noisew = 1, sigma2, 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-sided 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

`sigma2` variance of dispersion of observation noise.

`bw` shape parameter of observation noise (for `noisew` = 2).

`initd` type of density function.

1 :	Gaussian (normal)
2 :	uniform
3 :	two-sided exponential

k	number of intervals.
plot	logical. If TRUE (default), 'trend' and 'smt' are plotted.
...	further arguments to be passed to plot.ngsmth.

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

This broad family of distributions includes the Cauchy distribution ($b = 1$) and t-distribution ($b = (k+1)/2$).

Value

An object of class "ngsmth". It contains the following components:

trend	trend.
smt	smoothed density.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

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 <- x + rnorm(400, mean = 0, sd = 1.0)

# system noise density : Gaussian (normal)
s1 <- ngsmth(y, noisev = 1, tau2 = 1.4e-02, noisew = 2, sigma2 = 1.048)

plot(s1, "smt", theta = 20, phi = 60, expand = 0.3)

# system noise density : Pearson family
s2 <- ngsmth(y, noisev = 2, tau2 = 2.11e-10, bv = 0.6, noisew = 2,
             sigma2 = 1.042)
```

```
plot(s2, "smt", theta = 25, phi = 30, expand = 0.25)

## seismic data
data(MYE1F)
n <- length(MYE1F)
yy <- rep(0, n)
for (i in 2:n) yy[i] <- MYE1F[i] - 0.5 * MYE1F[i-1]
m <- seq(1, n, by = 2)
y <- yy[m]
z <- tvvar(y, trend.order = 2, tau2.ini = 4.909e-02, delta = 1.0e-06)

# system noise density : Gaussian (normal)
s3 <- ngsmth(z$sm, noisev = 1, tau2 = z$tau2, noisew = 2, sigma2 = pi*pi/6,
             k = 190)

plot(s3, "smt", phi = 50, expand = 0.5, col = 8)
```

---

pdfunc *Probability Density Function*

Description

Evaluate probability density function for normal distribution, Cauchy distribution, Pearson distribution, exponential distribution, Chi-square distributions, double exponential distribution and uniform distribution.

Usage

```
pdfunc(model = "norm", mean = 0, sigma2 = 1, mu = 0, tau2 = 1, shape,
       lambda = 1, side = 1, df, xmin = 0, xmax = 1, plot = TRUE, ...)
```

Arguments

`model`	a character string indicating the model type of probability density function: either `"norm"`, `"Cauchy"`, `"Pearson"`, `"exp"`, `"Chi2"`, `"dexp"` or `"unif"`.
`mean`	mean. (valid for `"norm"`)
`sigma2`	variance. (valid for `"norm"`)
`mu`	location parameter μ. (valid for `"Cauchy"` and `"Pearson"`)
`tau2`	dispersion parameter τ^2. (valid for `"Cauchy"` and `"Pearson"`)
`shape`	shape parameter (> 0). (valid for `"Pearson"`)
`lambda`	lambda λ. (valid for `"exp"`)
`side`	1: exponential, 2: two-sided exponential. (valid for `"exp"`)
`df`	degree of freedoms k. (valid for `"Chi2"`)
`xmin`	lower bound of the interval.
`xmax`	upper bound of the interval.
`plot`	logical. If `TRUE` (default), probability density function is plotted.
`...`	further arguments to be passed to `plot.pdfunc`.

Value

An object of class "pdfunc", which is a list with the following elements:

density	values of density function.
interval	lower and upper bound of interval.
param	parameters of model.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# normal distribution
pdfunc(model = "norm", xmin = -4, xmax = 4)

# Cauchy distribution
pdfunc(model = "Cauchy", xmin = -4, xmax = 4)

# Pearson distribution
pdfunc(model = "Pearson", shape = 2, xmin = -4, xmax = 4)

# exponential distribution
pdfunc(model = "exp", xmin = 0, xmax = 8)

pdfunc(model = "exp", xmin = -4, xmax = 4)

# Chi-square distribution
pdfunc(model = "Chi2", df = 3, xmin = 0, xmax = 8)

# double exponential distribution
pdfunc(model = "dexp", xmin = -4, xmax = 2)

# uniform distribution
pdfunc(model = "unif", xmin = 0, xmax = 1)
```

---

period *Compute a Periodogram*

Description

Compute a periodogram of the univariate time series.

Usage

```
period(y, window = 1, minmax = c(-1.0e+30, 1.0e+30), plot = TRUE, ...)
```

Arguments

`y`	a univariate time series.
`window`	smoothing window type. (0: box-car, 1: Hanning, 2: Hamming)
`minmax`	bound for outliers in low side and high side.
`plot`	logical. If `TRUE` (default), smoothed periodogram is plotted.
`...`	further arguments to be passed to `plot.spg`.

Details

$$\begin{array}{rll} \text{Hanning Window :} & W_0 = 0.5 & W_1 = 0.25 \\ \text{Hamming Window :} & W_0 = 0.54 & W_1 = 0.23 \end{array}$$

Value

An object of class `"spg"`, which is a list with the following elements:

`period`	periodogram (raw spectrum).
`smoothed.period`	smoothed periodogram. If there is not a negative number, logarithm of smoothed periodogram.
`log.scale`	if `TRUE` "smooth the periodogram on log scale.
`tsname`	the name of the univariate time series `y`.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# BLSALLFOOD data
data(BLSALLFOOD)
period(BLSALLFOOD)

# seismic Data
data(MYE1F)
period(MYE1F)
```

plot.arma — *Plot Analysis Result of ARMA Model*

Description

Plot impulse, autocovariance, PARCOR and characteristic roots of scalar ARMA model returned by `armaimp`.

Usage

```
## S3 method for class 'arma'
plot(x, ...)
```

Arguments

`x`	an object of class `"arma"` created by `armaimp`.
`...`	further graphical parameters may also be supplied as arguments.

plot.lsqr — *Plot Fitted Trigonometric Polynomial*

Description

Plot original data and fitted trigonometric polynomial returned by `lsqr`.

Usage

```
## S3 method for class 'lsqr'
plot(x, rdata = NULL, ...)
```

Arguments

`x`	an object of class `"lsqr"`.
`rdata`	original data, if necessary.
`...`	further graphical parameters may also be supplied as arguments.

plot.ngsmth *Plot Smoothed Density Function*

Description

Plot the smoothed density function returned by ngsmth.

Usage

```
## S3 method for class 'ngsmth'
plot(x, type = c("trend", "smt"), theta = 0, phi = 15,
                      expand = 1, col = "lightblue", ticktype= "detail", ...)
```

Arguments

x	an object of class "ngsmth".
type	plotted values, either or both of "trend" and "smt".
theta, phi, expand, col, ticktype	graphical parameters in perspective plot persp.
...	further graphical parameters may also be supplied as arguments.

plot.polreg *Plot Fitted Polynomial Trend*

Description

Plot trend component of fitted polynomial returned by polreg.

Usage

```
## S3 method for class 'polreg'
plot(x, rdata = NULL, ...)
```

Arguments

x	an object of class "polreg".
rdata	original data, if necessary.
...	further graphical parameters may also be supplied as arguments.

plot.season *Plot Trend, Seasonal and AR Components*

Description

Plot trend component, seasonal component, AR component and noise returned by `season`.

Usage

```
## S3 method for class 'season'
plot(x, rdata = NULL, ...)
```

Arguments

`x`	an object of class `"season"`.
`rdata`	original data, if necessary.
`...`	further graphical parameters may also be supplied as arguments.

plot.simulate *Plot Simulated Data Generated by State Space Model*

Description

Plot simulated data of Gaussian / non-Gaussian generated by state space model.

Usage

```
## S3 method for class 'simulate'
plot(x, use = NULL, ...)
```

Arguments

`x`	an object of class `"simulate"` created by `simssm` and `ngsim`.
`use`	start and end time `c(x1, x2)` to be plotted actually.
`...`	further graphical parameters may also be supplied as arguments.

plot.smooth *Plot Mean Vectors of Smoother*

Description

Plot Mean vectors of the smoother and standard deviation returned by tsmooth.

Usage

```
## S3 method for class 'smooth'
plot(x, rdata = NULL, ...)
```

Arguments

x	an object of class "smooth" created by tsmooth.
rdata	original data, if necessary.
...	further graphical parameters may also be supplied as arguments.

plot.spg *Plot Smoothed Periodogram*

Description

Plot smoothed periodogram or logarithm of smoothed periodogram.

Usage

```
## S3 method for class 'spg'
plot(x, ...)
```

Arguments

x	an object of class "spg" created by period and fftper.
...	further graphical parameters may also be supplied as arguments.

plot.trend *Plot Trend and Residuals*

Description

Plot trend component and residuals returned by `trend`.

Usage

```
## S3 method for class 'trend'
plot(x, rdata = NULL, ...)
```

Arguments

`x`	an object of class "trend".
`rdata`	original data, if necessary.
`...`	further graphical parameters may also be supplied as arguments.

plot.tvspc *Plot Evolutionary Power Spectra Obtained by Time Varying AR Model*

Description

Plot evolutionary power spectra obtained by time varying AR model returned by `tvspc`.

Usage

```
## S3 method for class 'tvspc'
plot(x, theta = 0, phi = 15, expand = 1, col = "lightblue",
        ticktype= "detail", ...)
```

Arguments

`x`	an object of class "tvspc".
`theta, phi, expand, col, ticktype`	graphical parameters in perspective plot `persp`.
`...`	further graphical parameters may also be supplied as arguments.

Examples

```
# seismic data
data(MYE1F)
z <- tvar(MYE1F, trend.order = 2, ar.order = 8, span = 20,
          outlier = c(630, 1026), tau2.ini = 6.6e-06, delta = 1.0e-06)
spec <- tvspc(z$arcoef, z$sigma2)
plot(spec, theta = 30, phi = 40, expand = 0.5)
```

polreg	*Polynomial Regression Model*

Description

Estimate the trend using the AIC best polynomial regression model.

Usage

```
polreg(y, order, plot = TRUE, ...)
```

Arguments

`y`	a univariate time series.
`order`	order of polynomial regression.
`plot`	logical. If `TRUE` (default), '`y`' and '`trend`' are plotted.
`...`	further arguments to be passed to plot.polreg.

Value

An object of class `"polreg"`, which is a list with the following elements:

`order.maice`	MAICE (minimum AIC estimate) order.
`sigma2`	residual variance of the model with order M. ($0 \leq M \leq$ `order` $+1$)
`aic`	AIC of the model with order M. ($0 \leq M \leq$ `order` $+1$)
`daic`	AIC - minimum AIC.
`coef`	regression coefficients $A(I, M)$ with order M. ($1 \leq M \leq$ `order` $+1$, $1 \leq I \leq M$)
`trend`	trend component.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# The daily maximum temperatures for Tokyo
data(Temperature)
polreg(Temperature, order = 7)

# Wholesale hardware data
data(WHARD)
y <- log10(WHARD)
polreg(y, order = 15)
```

season *Seasonal Adjustment*

Description

Seasonal adjustment by state space modeling.

Usage

```
season(y, trend.order = 1, seasonal.order = 1, ar.order = 0, trade = FALSE,
       period = 12, tau2.ini = NULL, filter = c(1, length(y)),
       predict = length(y), arcoef.ini = NULL, log = FALSE,
       minmax = c(-1.0e+30, 1.0e+30), plot = TRUE, ...)
```

Arguments

y	a univariate time series.
trend.order	trend order.
seasonal.order	seasonal order.
ar.order	AR order.
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.
period	number of seasons in one period. If the tsp attribute of y is not NULL, frequency(y). = 12 : for monthly data = 4 : for quarterly data
tau2.ini	initial estimate of variance of the system noise τ^2, not equal to 1.
filter	a numerical vector of the form c(x1,x2) which gives start and end position of filtering.
predict	the end position of prediction ($\geq$ x2).
arcoef.ini	initial estimate of AR coefficients (for ar.order > 0).
log	logical. If TRUE, the data y is log-transformed.
minmax	lower and upper limits of observations.
plot	logical. If TRUE (default), 'trend', 'seasonal' and 'ar' are plotted.
...	further arguments to be passed to plot.season.

Value

An object of class "season", which is a list with the following elements:

tau2	variance of the system noise.
sigma2	variance of the observational noise.

`llkhood`	log-likelihood of the model.
`aic`	AIC of the model.
`trend`	trend component (for `trend.order` > 0).
`seasonal`	seasonal component (for `seasonal.order` > 0).
`arcoef`	AR coefficients (for `ar.order` > 0).
`ar`	AR component (for `ar.order` > 0).
`day.effect`	trading day effect (for `trade` = 6).
`noise`	noise component.
`cov`	covariance matrix of smoother.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# BLSALLFOOD data
data(BLSALLFOOD)
season(BLSALLFOOD, trend.order = 2, seasonal.order = 1, ar.order = 2)

season(BLSALLFOOD, trend.order = 2, seasonal.order = 1, ar.order = 2,
       filter = c(1, 132))

# Wholesale hardware data
data(WHARD)
season(WHARD, trend.order = 2, seasonal.order = 1, ar.order = 0, trade = TRUE,
       log = TRUE)

season(WHARD, trend.order = 2, seasonal.order = 1, ar.order = 0, trade = TRUE,
       filter = c(1, 132), log = TRUE)
```

`simssm` *Simulation by Gaussian State Space Model*

Description

Simulate time series by Gaussian State Space Model.

Usage

```
simssm(n = 200, trend = NULL, seasonal.order = 0, seasonal = NULL,
       arcoef = NULL, ar = NULL, tau1 = NULL, tau2 = NULL, tau3 = NULL,
       sigma2 = 1.0, seed = NULL, plot = TRUE, ...)
```

Arguments

n	the number of simulated data.
trend	initial values of trend component of length at most 2.
seasonal.order	seasonal order. (0 or 1)
seasonal	if seasonal.order > 0, initial values of seasonal component of length $p - 1$, where p is the number of season in one period.
arcoef	AR coefficients.
ar	initial values of AR component.
tau1	variance of trend component model.
tau2	variance of AR component model.
tau3	variance of seasonal component model.
sigma2	variance of the observation noise.
seed	arbitrary positive integer to generate a sequence of uniform random numbers. The default seed is based on the current time.
plot	logical. If TRUE (default), simulated data are plotted.
...	further arguments to be passed to plot.simulate.

Value

An object of class "simulate", giving simulated data of Gaussian state space model.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# BLSALLFOOD data
data(BLSALLFOOD)
m1 <- 2; m2 <- 1; m3 <- 2
z <- season(BLSALLFOOD, trend.order = m1, seasonal.order = m2, ar.order = m3)

nl <- length(BLSALLFOOD)
trend <- z$trend[m1:1]
arcoef <- z$arcoef
period <- 12
seasonal <- z$seasonal[(period-1):1]
ar <- z$ar[m3:1]
tau1 <- z$tau2[1]
tau2 <- z$tau2[2]
tau3 <- z$tau2[3]
simssm(n = nl, trend, seasonal.order = m2, seasonal, arcoef, ar, tau1, tau2,
       tau3, sigma2 = z$sigma2, seed = 333)
```

`Sunspot` *Sun Spot Number Data*

Description

Yearly numbers of sunspots from to 1749 to 1979.

Usage

```
data(Sunspot)
```

Format

A time series of 231 observations; yearly from 1749 to 1979.

Details

Sunspot is a part of the dataset `sunspot.year` from 1700 to 1988. Value "0" is converted into "0.1" for log transformation.

`Temperature` *Temperatures Data*

Description

The daily maximum temperatures for Tokyo (from 1979-01-01 to 1980-04-30).

Usage

```
data(Temperature)
```

Format

A time series of 486 observations.

Source

The data were obtained from Tokyo District Meteorological Observatory.

http://www.data.jma.go.jp/obd/stats/etrn/

`trend` *Trend Estimation*

Description

Estimate the trend by state space model.

Usage

```
trend(y, trend.order = 1, tau2.ini = NULL, delta, plot = TRUE, ...)
```

Arguments

`y`	a univariate time series.
`trend.order`	trend order.
`tau2.ini`	initial estimate of variance of the system noise τ^2. If `tau2.ini` = `NULL`, the most suitable value is chosen in $\tau^2 = 2^{-k}$.
`delta`	search width (for `tau2.ini` is specified (not `NULL`)) .
`plot`	logical. If `TRUE` (default), '`trend`' and '`residual`' are plotted.
`...`	further arguments to be passed to plot.trend.

Details

The trend model can be represented by a state space model

$$x_n = Fx_{n-1} + Gv_n,$$

$$y_n = Hx_n + w_n,$$

where F, G and H are matrices with appropriate dimensions. We assume that v_n and w_n are white noises that have the normal distributions $N(0, \tau^2)$ and $N(0, \sigma^2)$, respectively.

Value

An object of class `"trend"`, which is a list with the following elements:

`trend`	trend component.
`residual`	residuals.
`tau2`	variance of the system noise τ^2.
`sigma2`	variance of the observational noise σ^2.
`llkhood`	log-likelihood of the model.
`aic`	AIC.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# The daily maximum temperatures for Tokyo
data(Temperature)
trend(Temperature, trend.order = 1, tau2.ini = 0.223, delta = 0.001)

trend(Temperature, trend.order = 2)
```

`tsmooth` *Prediction and Interpolation of Time Series*

Description

Predict and interpolate time series based on state space model by Kalman filter.

Usage

```
tsmooth(y, f, g, h, q, r, x0 = NULL, v0 = NULL, filter.end = NULL,
        predict.end = NULL, minmax = c(-1.0e+30, 1.0e+30), missed = NULL,
        np = NULL, plot = TRUE, ...)
```

Arguments

`y`	a univariate time series y_n.
`f`	state transition matrix F_n.
`g`	matrix G_n.
`h`	matrix H_n.
`q`	system noise variance Q_n.
`r`	observational noise variance R.
`x0`	initial state vector $X(0 \mid 0)$.
`v0`	initial state covariance matrix $V(0 \mid 0)$.
`filter.end`	end point of filtering.
`predict.end`	end point of prediction.
`minmax`	lower and upper limits of observations.
`missed`	start position of missed intervals.
`np`	number of missed observations.
`plot`	logical. If `TRUE` (default), '`mean.smooth`' and '`esterr`' are plotted.
`...`	further arguments to be passed to `plot.smooth`.

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 a univariate time series, x_n is an m-dimensional state vector.

F_n, G_n and H_n are $m \times m$, $m \times k$ matrices and a vector of length m , respectively. Q_n is $k \times k$ matrix and R_n is a scalar. v_n is system noise and w_n is observation noise, where we assume that $E(v_n, w_n) = 0$, $v_n \sim N(0, Q_n)$ and $w_n \sim N(0, R_n)$. 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

An object of class `"smooth"`. It contains the following components:

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

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

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

Examples

```
## Example of prediction (AR model : m=15, k=1)
data(BLSALLFOOD)
BLS120 <- BLSALLFOOD[1:120]
z1 <- arfit(BLS120, plot = FALSE)
tau2 <- z1$sigma2
arcoef <- z1$arcoef

# in case m = 15
m1 <- z1$maice.order
f <- matrix(0.0e0, m1, m1)
f[1, ] <- arcoef[1:m1]
if (m1 != 1)
  for (i in 2:m1) f[i, i-1] <- 1
g <- c(1, rep(0.0e0, m1-1))
h <- c(1, rep(0.0e0, m1-1))
q <- tau2[m1+1]
r <- 0.0e0
x0 <- rep(0.0e0, m1)
```

```
v0 <- NULL

s1 <- tsmooth(BLS120, f, g, h, q, r, x0, v0, filter.end = 120, predict.end = 156)
s1

plot(s1, BLSALLFOOD)

## Example of interpolation of missing values  (AR model : m=15, k=1)
z2 <- arfit(BLSALLFOOD, plot = FALSE)
tau2 <- z2$sigma2
arcoef <- z2$arcoef

# in case m2 = 15
m2 <- z2$maice.order
f <- matrix(0.0e0, m2, m2)
f[1, ] <- arcoef[1:m2]
if (m2 != 1)
  for (i in 2:m2) f[i, i-1] <- 1
g <- c(1, rep(0.0e0, m2-1))
h <- c(1, rep(0.0e0, m2-1))
q <- tau2[m2+1]
r <- 0.0e0
x0 <- rep(0.0e0, m2)
v0 <- NULL

tsmooth(BLSALLFOOD, f, g, h, q, r, x0, v0, missed = c(41, 101), np = c(30, 20))
```

---

tvar *Time Varying Coefficients AR Model*

Description

Estimate time varying coefficients AR model.

Usage

```
tvar(y, trend.order = 2, ar.order = 2, span, outlier = NULL, tau2.ini = NULL,
     delta, plot = TRUE)
```

Arguments

`y`	a univariate time series.
`trend.order`	trend order (1 or 2).
`ar.order`	AR order.
`span`	local stationary span.
`outlier`	positions of outliers.
`tau2.ini`	initial estimate of variance of the system noise τ^2. If `tau2.ini` = `NULL`, the most suitable value is chosen in $\tau^2 = 2^{-k}$.

`delta`	search width.
`plot`	logical. If `TRUE` (default), '`parcor`' is plotted.

Details

The time-varying coefficients AR model is given by

$$y_t = a_{1,t} y_{t-1} + \ldots + a_{p,t} y_{t-p} + u_t$$

where $a_{i,t}$ is i-lag AR coefficient at time t and u_t is a zero mean white noise.

The time-varying spectrum can be plotted using AR coefficient `arcoef` and variance of the observational noise `sigma2` by plot.tvspc (see tvspc).

Value

`arcoef`	time varying AR coefficients.
`sigma2`	variance of the observational noise σ^2.
`tau2`	variance of the system noise τ^2.
`llkhood`	log-likelihood of the model.
`aic`	AIC.
`parcor`	partial autocorrelation coefficient.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

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 Control, AC-30, 48-56.

Examples

```
# seismic data
data(MYE1F)
z <- tvar(MYE1F, trend.order = 2, ar.order = 8, span = 20,
          outlier = c(630, 1026), tau2.ini = 6.6e-06, delta = 1.0e-06)
z

spec <- tvspc(z$arcoef, z$sigma2)
plot(spec, theta = 30, phi = 40, expand = 0.5)
```

tvspc *Evolutionary Power Spectra by Time Varying AR Model*

Description

Estimate evolutionary power spectra by time varying AR model.

Usage

```
tvspc(arcoef, sigma2, var = NULL, span = 20, nf = 200)
```

Arguments

arcoef	time varying AR coefficients.
sigma2	variance of the observational noise.
var	time varying variance.
span	local stationary span.
nf	number of frequencies in evaluating spectrum.

Value

return an object of class "tvspc".

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

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 Control, AC-30, 48-56.

Examples

```
# seismic data
data(MYE1F)
z <- tvar(MYE1F, trend.order = 2, ar.order = 8, span = 20,
          outlier = c(630, 1026), tau2.ini = 6.6e-06, delta = 1.0e-06)
spec <- tvspc(z$arcoef, z$sigma2)
plot(spec, theta = 30, phi = 40, expand = 0.5)
```

tvvar *Time Varying Variance*

Description

Estimate time-varying variance.

Usage

```
tvvar(y, trend.order, tau2.ini = NULL, delta, plot = TRUE, ...)
```

Arguments

`y`	a univariate time series.
`trend.order`	trend order.
`tau2.ini`	initial estimate of variance of the system noise τ^2. If `tau2.ini` = `NULL`, the most suitable value is chosen in $\tau^2 = 2^{-k}$.
`delta`	search width.
`plot`	logical. If `TRUE` (default), '`sm`', '`trend`' and '`noise`' are plotted.
`...`	further arguments to be passed to `plot.tvvar`.

Details

Assuming that $\sigma^2_{2m-1} = \sigma^2_{2m}$, we define a transformed time series $s_1, \ldots, s_{N/2}$ by

$$s_m = y^2_{2m-1} + y^2_{2m},$$

where y_n is a Gaussian white noise with mean 0 and variance σ^2_n. s_m is distributed as a χ^2 distribution with 2 degrees of freedom, so the probability density function of s_m is given by

$$f(s) = \frac{1}{2\sigma^2} e^{-s/2\sigma^2}.$$

By further transformation

$$z_m = \log\left(\frac{s_m}{2}\right),$$

the probability density function of z_m is given by

$$g(z) = \frac{1}{\sigma^2} \exp\left\{z - \frac{e^z}{\sigma^2}\right\} = \exp\left\{(z - \log\sigma^2) - e^{(z-\log\sigma^2)}\right\}.$$

Therefore, the transformed time series is given by

$$z_m = \log\sigma^2 + w_m,$$

where w_m is a double exponential distribution with probability density function

$$h(w) = \exp\{w - e^w\}.$$

In the space state model

$$z_m = t_m + w_m$$

by identifying trend components of z_m, the log variance of original time series y_n is obtained.

Value

An object of class `"tvvar"`, which is a list with the following elements:

`tvv`	time varying variance.
`nordata`	normalized data.
`sm`	transformed data.
`trend`	trend.
`noise`	residuals.
`tau2`	variance of the system noise.
`sigma2`	variance of the observational noise.
`llkhood`	log-likelihood of the model.
`aic`	AIC.
`tsname`	the name of the univariate time series `y`.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

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 Control, AC-30, 48-56.

Examples

```
# seismic data
data(MYE1F)
tvvar(MYE1F, trend.order = 2, tau2.ini = 6.6e-06, delta = 1.0e-06)
```

unicor *Autocovariance and Autocorrelation*

Description

Compute autocovariance and autocorrelation function of the univariate time series.

Usage

```
unicor(y, lag = NULL, minmax = c(-1.0e+30, 1.0e+30), 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.
minmax	bound for outliers in low side and high side.
plot	logical. If TRUE (default), autocorrelations are plotted.
...	further arguments to be passed to plot.unicor.

Value

An object of class "unicor", which is a list with the following elements:

acov	autocovariances.
acor	autocorrelations.
acov.err	error bound for autocovariances.
acor.err	error bound for autocorrelations.
mean	mean of y.
tsname	the name of the univariate time series y.

References

Kitagawa, G. (2010) *Introduction to Time Series Modeling*. Chapman & Hall/CRC.

Examples

```
# Yaw rate, rolling, pitching and rudder angle of a ship
data(HAKUSAN)
Yawrate <- HAKUSAN[, 1]
unicor(Yawrate, lag = 50)

# seismic data
data(MYE1F)
unicor(MYE1F, lag = 50)
```

WHARD *Wholesale Hardware Data*

Description

The monthly record of wholesale hardware data. (January 1967 - November 1979)

Usage

```
data(WHARD)
```

Format

A time series of 155 observations.

Source

The data were obtained from the United States Bureau of Labor Statistics (BLS).

Index