

# Package ‘TSSS’

September 29, 2023

**Version** 1.3.4-5

**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 (>= 3.6), 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  
(2020, ISBN: 978-0-367-18733-0).

**License** GPL (>= 2)

**Contact** Bug report mailing list <ismrp@grp.ism.ac.jp>

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2023-09-29 04:10:02 UTC

## R topics documented:

TSSS-package . . . . .	3
arfit . . . . .	3
armachar . . . . .	5
armafit . . . . .	6
armafit2 . . . . .	7
BLSALLFOOD . . . . .	8
boxcox . . . . .	8
crscor . . . . .	9
fftper . . . . .	10
Haibara . . . . .	12
HAKUSAN . . . . .	13
klinfo . . . . .	14
lsar . . . . .	15

lsar.chgpt . . . . .	16
lsqr . . . . .	17
marfit . . . . .	18
marlsq . . . . .	19
marspc . . . . .	20
MYE1F . . . . .	21
ngsim . . . . .	21
ngsmth . . . . .	23
Nikkei225 . . . . .	25
NLmodel . . . . .	26
pdfunc . . . . .	26
period . . . . .	28
pfilter . . . . .	29
pfilterNL . . . . .	32
PfilterSample . . . . .	34
plot.boxcox . . . . .	34
plot.lsqr . . . . .	35
plot.ngsmth . . . . .	35
plot.polreg . . . . .	36
plot.season . . . . .	36
plot.simulate . . . . .	37
plot.smooth . . . . .	37
plot.spg . . . . .	38
plot.trend . . . . .	38
plot.tvspc . . . . .	39
polreg . . . . .	39
Rainfall . . . . .	40
season . . . . .	41
simssm . . . . .	43
Sunspot . . . . .	44
Temperature . . . . .	45
trend . . . . .	45
tsmooth . . . . .	46
tvar . . . . .	49
tvspc . . . . .	50
tvvar . . . . .	51
unicor . . . . .	53
WHARD . . . . .	54
<b>Index</b>	<b>55</b>

---

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".

After that, the revised edition "Introduction to Time Series Analysis (in Japanese)" and the translation version "Introduction to Time Series Modeling" are published.

Currently the revised edition "Introduction to Time Series Modeling with Applications in R" is published, in which calculations of most of the modeling or methods are explained using this package.

### References

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

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

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

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

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

Kitagawa, G. (2020) *Introduction to Time Series Modeling with R*. Iwanami Publishing Company (in Japanese).

---

arfit

*Univariate AR Model Fitting*

---

### Description

Fit a univariate AR model by the Yule-Walker method, the least squares (Householder) method or the 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. <ul style="list-style-type: none"> <li>1 : Yule-Walker method</li> <li>2 : Least squares (Householder) method</li> <li>3 : PARCOR method (Partial autoregression)</li> <li>4 : PARCOR method (PARCOR)</li> <li>5 : PARCOR method (Burg's algorithm)</li> </ul>
plot	logical. If TRUE (default), PARCOR, AIC and power spectrum are plotted.
...	graphical arguments passed to the plot method.

**Value**

An object of class "arfit" which has a plot method. This is a list with the following components:

sigma2	innovation variance.
maice.order	order of minimum AIC.
aic	AICs of the estimated AR models.
arcoef	AR coefficients of the estimated AR models.
parcor	PARCOR.
spec	power spectrum (in log scale) of the AIC best AR model.
tname	the name of the univariate time series $y$ .

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

**Examples**

```
# Sunspot number data
data(Sunspot)
arfit(log10(Sunspot), lag = 20, method = 1)

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

armachar

*Calculate Characteristics of Scalar ARMA Model***Description**

Calculate impulse response function, autocovariance function, autocorrelation function and characteristic roots of given scalar ARMA model.

**Usage**

```
armachar(arcoef = NULL, macoef = NULL, v, lag = 50, nf = 200, plot = TRUE, ...)
```

**Arguments**

arcoef	AR coefficients.
macoef	MA coefficients.
v	innovation variance.
lag	maximum lag of autocovariance function.
nf	number of frequencies in evaluating spectrum.
plot	logical. If TRUE (default), impulse response function, autocovariance, power spectrum, PARCOR and characteristic roots are plotted.
...	graphical arguments passed to the plot method.

**Details**

The ARMA model is given by

$$y_t - a_1 y_{t-1} - \dots - a_p y_{t-p} = u_t - b_1 u_{t-1} - \dots - b_q u_{t-q},$$

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 has a plot method. This is a list with components:

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

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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)
armachar(arcoef = a, v = 1.0, 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)
armachar(macoeef = b, v = 1.0, lag = 20)

# ARMA model :  $y(n) = a(1)y(n-1) + a(2)y(n-2)$ 
#                +  $v(n) - b(1)v(n-1) - b(2)v(n-2)$ 
armachar(arcoef = a, macoeef = b, v = 1.0, lag = 20)
```

---

 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. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

**Examples**

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

armachar(arcoef = z$arcoef, macoef = z$macoef, v = z$sigma2, lag = 20)
```

armafit2

*Scalar ARMA Model Fitting***Description**

Estimate all ARMA models within the user-specified maximum order by maximum likelihood method.

**Usage**

```
armafit2(y, ar.order, ma.order)
```

**Arguments**

y	a univariate time series.
ar.order	maximum AR order.
ma.order	maximum MA order.

**Value**

aicmin	minimum AIC.
maice.order	AR and MA orders of minimum AIC model.
sigma2	innovation variance of all models.
llkhood	log-likelihood of all models.
aic	AIC of all models.
coef	AR and MA coefficients of all models.

## References

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

## Examples

```
# Sunspot number data
data(Sunspot)
y <- log10(Sunspot)
armafit2(y, ar.order = 5, ma.order = 5)
```

---

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

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

## Usage

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



**Arguments**

<code>y</code>	a univariate time series.
<code>plot</code>	logical. If TRUE (default), original data and transformed data with minimum AIC are plotted.
<code>...</code>	graphical arguments passed to <code>plot.bboxcox</code> .

**Value**

An object of class "bboxcox", which is a list with the following components:

<code>mean</code>	mean of original data.
<code>var</code>	variance of original data.
<code>aic</code>	AIC of the model with respect to the original data.
<code>llkhood</code>	log-likelihood of the model with respect to the original data.
<code>z</code>	transformed data with the AIC best lambda.
<code>aic.z</code>	AIC of the model with respect to the transformed data.
<code>llkhood.z</code>	log-likelihood of the model with respect to the transformed data.

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

**Examples**

```
# Sunspot number data
data(Sunspot)
bboxcox(Sunspot)

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

---

crscor

*Cross-Covariance and Cross-Correlation*


---

**Description**

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

**Usage**

```
crscor(y, lag = NULL, outmin = NULL, outmax = NULL, plot = TRUE, ...)
```

**Arguments**

<code>y</code>	a multivariate time series.
<code>lag</code>	maximum lag. Default is $2\sqrt{n}$ , where $n$ is the length of the time series $y$ .
<code>outmin</code>	bound for outliers in low side. A default value is $-1.0e+30$ for each dimension.
<code>outmax</code>	bound for outliers in high side. A default value is $1.0e+30$ for each dimension.
<code>plot</code>	logical. If TRUE (default), cross-correlations are plotted.
<code>...</code>	graphical arguments passed to the <code>plot</code> method.

**Value**

An object of class "crscor" which has a `plot` method. This is a list with the following components:

<code>cov</code>	cross-covariances.
<code>cor</code>	cross-correlations.
<code>mean</code>	mean vector.

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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)

# The groundwater level and the atmospheric pressure
data(Haibara)
crscor(Haibara, 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**

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

**Details**

Hanning Window :  $W_0 = 0.5$      $W_1 = 0.25$   
 Hamming Window :  $W_0 = 0.54$      $W_1 = 0.23$

**Value**

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

<code>period</code>	periodogram.
<code>smoothed.period</code>	smoothed periodogram. If there is not a negative number, logarithm of smoothed periodogram.
<code>log.scale</code>	logical. If TRUE <code>smoothed.period</code> is logarithm of smoothed periodogram.
<code>tsname</code>	the name of the univariate time series <code>y</code> .

**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. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

**Examples**

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

---

Haibara

*Haibara Data*

---

### Description

A bivariate time series of the groundwater level and the atmospheric pressure that were observed at 10-minute intervals at the Haibara observatory of the Tokai region, Japan.

### Usage

```
data(Haibara)
```

### Format

A data frame with 400 observations on the following 2 variables.

```
[, 1] Groundwater level  
[, 2] Atmospheric pressure
```

### Source

The data were offered by Dr. M. Takahashi and Dr. N. Matsumoto of National Institute of Advanced Industrial Science and Technology.

### Examples

```
data(Haibara)  
  
## put histograms on the diagonal  
panel.hist <- function(x, ...)  
{  
  usr <- par("usr")  
  par(usr = c(usr[1:2], 0, 1.3))  
  nB <- 15; nB1 <- nB + 1  
  xmin <- min(x, na.rm = TRUE)  
  xmax <- max(x, na.rm = TRUE)  
  w <- (xmax - xmin) / nB  
  breaks <- xmin  
  b <- xmin  
  for (i in 1:nB) {  
    b <- b + w  
    breaks <- c(breaks, b)  
  }  
  h <- hist(x, breaks = breaks, plot = FALSE)  
  y <- h$counts  
  y <- y / max(y)  
  rect(breaks[1:nB], 0, breaks[2:nB1], y, ...)  
}
```

```
par(xaxs = "i", yaxs = "i", xaxt = "n", yaxt = "n")
pairs(Haibara, diag.panel = panel.hist, pch = 20, cex.labels = 1.5,
      label.pos = 0.9, lower.panel = NULL)
```

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.

**Examples**

```
data(HAKUSAN)
HAKUSAN234 <- HAKUSAN[, c(2,3,4)]

## put histograms on the diagonal
panel.hist <- function(x, ...)
{
  usr <- par("usr")
  par(usr = c(usr[1:2], 0, 1.3))
  nB <- 20; nB1 <- nB + 1
  xmin <- min(x)
  xmax <- max(x)
  w <- (xmax - xmin) / nB
  breaks <- xmin
  b <- xmin
  for (i in 1:nB) {
    b <- b + w
    breaks <- c(breaks, b)
  }
}
```

```

h <- hist(x, breaks = breaks, plot = FALSE)
y <- h$counts; y <- y / max(y)
rect(breaks[1:nB], 0, breaks[2:nB1], y, ...)

}

par(xaxis = "i", yaxis = "i", xaxt = "n", yaxt = "n")
pairs(HAKUSAN234, diag.panel = panel.hist, pch = 20, cex.labels = 1.5,
      label.pos = 0.9, lower.panel = NULL)

```

---

klinfo

*Kullback-Leibler Information*


---

### Description

Compute 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): $\mu$ (location parameter) paramg(2): $\tau^2$ (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): $\mu$ (location parameter) paramf(2): $\tau^2$ (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 $[-x_{\max}, x_{\max}]$ .

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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	length of basic local span.
plot	logical. If TRUE (default), local spectra are plotted.
...	graphical arguments passed to the plot method.

**Value**

An object of class "lsar" which has a plot method. This is a list with the following components:

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 in computing local power spectrum.
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. (2020) *Introduction to Time Series Modeling with Applications in R*. 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 $c(n_0, n_e)$ which gives a start and end point of time interval used for model fitting.
candidate	a vector of the form $c(n_1, n_2)$ which gives minimum and maximum of the candidate for change point. $n_0+2k < n_1 < n_2+k < n_e$ , ( $k$ is <code>max.arorder</code> )
plot	logical. If TRUE (default), $y[n_0:n_e]$ and <code>aic</code> are plotted.
...	graphical arguments passed to the plot method.



**Value**

An object of class "chgpt" which has a plot method. This is a list with the following components:

aic	AICs of the AR models fitted on [n1, n2].
aicmin	minimum AIC.
change.point	estimated change point.
subint	information about the original sub-interval.

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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 by the least squares method via Householder transformation.

**Usage**

```
lsqr(y, lag = NULL, period = 365, plot = TRUE, ...)
```

**Arguments**

y	a univariate time series.
lag	number of sine and cosine components. Default is $\sqrt{n}$ , where $n$ is the length of the time series $y$ .
period	period of one cycle.
plot	logical. If TRUE (default), original data and fitted trigonometric polynomial are plotted.
...	graphical arguments passed to <a href="#">plot.lsqr</a> .

**Value**

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

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

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

**Examples**

```
# The daily maximum temperatures in Tokyo
data(Temperature)
lsqr(Temperature, lag = 10)
```

---

marfit

*Yule-Walker Method of Fitting Multivariate AR Model*


---

**Description**

Fit a multivariate AR model by the 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 components:

maice.order	order of minimum AIC.
aic	AIC's of the AR models with order $0, \dots, \text{lag}$ .
v	innovation covariance matrix of the AIC best model.
arcoef	AR coefficients of the AIC best model.

## References

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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 components:

maice.order	order of the MAICE model.
aic	AIC of the MAR model with minimum AIC orders.
v	innovation covariance matrix.
arcoef	AR coefficient matrices.

## References

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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, coherency 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, coherency and power contribution are plotted.
...	graphical arguments passed to the plot method.

**Value**

An object of class "marspc" which has a plot method. This is a list with the following components:

spec	cross spectra.
amp	amplitude spectra.
phase	phase spectra.
coh	simple coherency.
power	decomposition of power spectra.
rpower	relative power contribution.

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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 = NULL, paramw = NULL, noisev = 1,
      vminmax = NULL, paramv = NULL, seed = NULL, plot = TRUE, ...)
```

**Arguments**

n	number of data generated by simulation.
trend	initial values of trend component of length $m1$ , where $m1$ is trend order (1, 2). If NULL (default), trend order is 0.
seasonal.order	order of seasonal component model (0, 1, 2).
seasonal	if seasonal.order > 0, initial values of seasonal component of length $p - 1$ , where $p$ is period of one season.
arcoef	AR coefficients.
ar	initial values of AR component.
noisew	type of the observational noise.  -1 : Cauchy random number -2 : exponential distribution -3 : double exponential distribution 0 : double exponential distribution (+ Euler's constant) 1 : normal distribution (generated by inverse function) 2 : Pearson distribution (generated by inverse function) 3 : double exponential distribution (generated by inverse function)
wminmax	lower and upper bound of observational noise.
paramw	parameter of the observational noise density.  noisew = 1 : variance noisew = 2 : dispersion parameter (tau square) and shape parameter
noisev	type of the system noise.  -1 : Cauchy random number -2 : exponential distribution -3 : double exponential distribution 0 : double exponential distribution (+ Euler's constant) 1 : normal distribution (generated by inverse function) 2 : Pearson distribution (generated by inverse function) 3 : double exponential distribution (generated by inverse function)
vminmax	lower and upper bound of system noise.
paramv	parameter of the system noise density.  noisev = 1 : variance noisev = 2 : dispersion parameter (tau square) and 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.
...	graphical arguments passed to <a href="#">plot.simulate</a> .

**Value**

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

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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. <ol style="list-style-type: none"> <li>1: Gaussian (normal)</li> <li>2: Pearson family</li> <li>3: two-sided exponential</li> </ol>
tau2	variance or dispersion of system noise.
bv	shape parameter of system noise (for noisev = 2).
noisew	type of observation noise density <ol style="list-style-type: none"> <li>1: Gaussian (normal)</li> <li>2: Pearson family</li> </ol>

	3 : two-sided exponential
	4 : double exponential
sigma2	variance or 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 in numerical integration.
plot	logical. If TRUE (default), trend is plotted.
...	graphical arguments passed to <code>plot.ngsmth</code> .

### 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  $c = \tau^{2b-1}\Gamma(b) / \Gamma(\frac{1}{2})\Gamma(b - \frac{1}{2})$ .

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

### Value

An object of class "ngsmth", which is a list with the following components:

llkhood	log-likelihood.
trend	trend.
smt	smoothed density.

### References

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

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



**Examples**

```
## test data
data(PfilterSample)
par(mar = c(3, 3, 1, 1) + 0.1)

# system noise density : Gaussian (normal)
s1 <- ngsmth(PfilterSample, noisev = 1, tau2 = 1.4e-02, noisew = 1, sigma2 = 1.048)
s1
plot(s1, "smt", theta = 25, phi = 30, expand = 0.25, col = "white")

# system noise density : Pearson family
s2 <- ngsmth(PfilterSample, noisev = 2, tau2 = 2.11e-10, bv = 0.6, noisew = 1,
             sigma2 = 1.042)
s2
plot(s2, "smt", theta = 25, phi = 30, expand = 0.25, col = "white")

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

---

Nikkei225

*Nikkei225*


---

**Description**

A daily closing values of the Japanese stock price index, Nikkei225, quoted from January 4, 1988, to December 30, 1993.

**Usage**

```
data(Nikkei225)
```

**Format**

A time series of 1480 observations.

**Source**

<https://indexes.nikkei.co.jp/nkave/archives/data>

---

 NLmodel

*The Nonlinear State-Space Model Data*


---

**Description**

The series generated by the nonlinear state-space model.

**Usage**

data(NLmodel)

**Format**

A matrix with 100 rows and 2 columns.

$$\begin{array}{ll} [, 1] & x_n \\ [, 2] & y_n \end{array}$$
**Details**

The system model  $x_n$  and the observation model  $y_n$  are generated by following state-space model:

$$x_n = \frac{1}{2}x_{n-1} + \frac{25x_{n-1}}{x_{n-1}^2 + 1} + 8\cos(1.2n) + v_n$$

$$y_n = \frac{x_n^2}{10} + w_n,$$

where  $v_n \sim N(0, 1)$ ,  $w_n \sim N(0, 10)$ ,  $v_0 \sim N(0, 5)$ .

---

 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 $\mu$ . (valid for "Cauchy" and "Pearson")
tau2	dispersion parameter $\tau^2$ . (valid for "Cauchy" and "Pearson")
shape	shape parameter ( $> 0.5$ ). (valid for "Pearson")
lambda	lambda $\lambda$ . (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.
...	graphical arguments passed to the plot method.

**Value**

An object of class "pdfunc" which has a plot method. This is a list with the following components:

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

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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, lag = NULL, 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)
lag	maximum lag of autocovariance. If NULL (default), window = 0 : lag = $n - 1$ , window > 0 : lag = $2\sqrt{n}$ , where $n$ is the length of data.
minmax	bound for outliers in low side and high side.
plot	logical. If TRUE (default), smoothed periodogram is plotted.
...	graphical arguments passed to <a href="#">plot.spg</a> .

### Details

Hanning Window :	$W_0 = 0.5$	$W_1 = 0.25$
Hamming Window :	$W_0 = 0.54$	$W_1 = 0.23$

### Value

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

period	periodogram(or raw spectrum).
--------	-------------------------------

smoothed.period smoothed log-periodogram. Smoothed periodogram is given if there is a negative value in the smoothed periodogram.

log.scale if TRUE "smooth the periodogram on log scale.

tsname the name of the univariate time series y.

## References

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

## Examples

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

## seismic Data
data(MYE1F)

# smoothed periodogram
period(MYE1F)

# periodogram
period(MYE1F, window = 0)

# raw spectrum
period(MYE1F, window = 0, lag = 200)

# Hamming window
period(MYE1F, window = 2)
```

---

pfilter

*Particle Filtering and Smoothing*

---

## Description

Trend estimation by particle filter and smoother.

## Usage

```
pfilter(y, m = 10000, model = 0, lag = 20, initd = 0, sigma2, tau2,
        alpha = 0.99, bigtau2 = NULL, init.sigma2 = 1, xrange = NULL,
        seed = NULL, plot = TRUE, ...)
```

**Arguments**

y	univariate time series.
m	number of particles.
model	model for the system noise.

	<ul style="list-style-type: none"> <li>0: normal distribution</li> <li>1: Cauchy distribution</li> <li>2: Gaussian mixture distribution  <math>\alpha N(0, \tau^2) + (1 - \alpha) N(0, T^2)</math>,            where <math>N</math> is the normal density.</li> </ul>
lag	lag length for fixed-lag smoothing.
initd	type of initial state distribution. <ul style="list-style-type: none"> <li>0: normal distribution</li> <li>1: uniform distribution</li> <li>2: Cauchy distribution</li> <li>3: fixed point (default value = 0)</li> </ul>
sigma2	observation noise variance $\sigma^2$ .
tau2	system noise variance $\tau^2$ for model = 0 or dispersion parameter for model = 1.
alpha	mixture weight $\alpha$ . (valid for model = 2)
bigtau2	variance of the second component $T^2$ . (valid for model = 2)
init.sigma2	variance for initd = 0 or dispersion parameter of initial state distribution for initd = 2.
xrange	specify the lower and upper bounds of the distribution's range.
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), marginal smoothed distribution is plotted.
...	graphical arguments passed to the plot method.

### Details

This function performs particle filtering and smoothing for the first order trend model;

$$\begin{aligned} x_n &= x_{n-1} + v_n, & (\text{system model}) \\ y_n &= x_n + w_n, & (\text{observation model}) \end{aligned}$$

where  $y_n$  is a time series,  $x_n$  is the state vector. The system noise  $v_n$  and the observation noise  $w_n$  are assumed to be white noises which follow a Gaussian distribution or a Cauchy distribution, and non-Gaussian distribution, respectively.

The algorithm of the particle filter and smoother are presented in Kitagawa (2020). For more details, please refer to Kitagawa (1996) and Doucet et al. (2001).

### Value

An object of class "pfilter" which has a plot method. This is a list with the following components:

llkhood	log-likelihood.
---------	-----------------

smooth.dist      marginal smoothed distribution of the trend  $T(i, j)$  ( $i = 1, \dots, n, j = 1, \dots, 7$ ), where  $n$  is the length of  $y$ .

j = 4:      50% point  
 j = 3, 5:    1-sigma points (15.87% and 84.14% points)  
 j = 2, 6:    2-sigma points (2.27% and 97.73% points)  
 j = 1, 7:    3-sigma points (0.13% and 99.87% points)

## References

Kitagawa, G. (1996) *Monte Carlo filter and smoother for non-Gaussian nonlinear state space models*, J. of Comp. and Graph. Statist., 5, 1-25.

Doucet, A., de Freitas, N. and Gordon, N. (2001) *Sequential Monte Carlo Methods in Practice*, Springer, New York.

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

## See Also

[pfilterNL](#) performs particle filtering and smoothing for nonlinear non-Gaussian state-space model.

## Examples

```
data(PfilterSample)
y <- PfilterSample

## Not run:
pfilter(y, m = 100000, model = 0, lag = 20, initd = 0, sigma2 = 1.048,
        tau2 = 1.4e-2, xrange = c(-4, 4), seed = 2019071117)

pfilter(y, m = 100000, model = 1, lag = 20, initd = 0, sigma2 = 1.045,
        tau2 = 3.53e-5, xrange = c(-4, 4), seed = 2019071117)

pfilter(y, m = 100000, model = 2, lag = 20, initd = 0, sigma2 = 1.03,
        tau2 = 0.00013, alpha = 0.991, xrange = c(-4, 4), seed = 2019071117)

## End(Not run)
```

---

pfilterNL

*Particle Filtering and Smoothing for Nonlinear State-Space Model*

---

## Description

Trend estimation by particle filter and smoother via nonlinear state-space model.

## Usage

```
pfilterNL(y, m = 10000, lag = 20, sigma2, tau2, xrange = NULL, seed = NULL,
          plot = TRUE, ...)
```



**Arguments**

y	univariate time series.
m	number of particles.
lag	lag length for fixed-lag smoothing.
sigma2	observation noise variance.
tau2	system noise variance.
xrange	specify the lower and upper bounds of the distribution's range.
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), marginal smoothed distribution is plotted.
...	graphical arguments passed to the plot method.

**Details**

This function performs particle filtering and smoothing for the following nonlinear state-space model;

$$x_n = \frac{1}{2}x_{n-1} + \frac{25x_{n-1}}{x_{n-1}^2+1} + 8\cos(1.2n) + v_n, \quad (\text{system model})$$

$$y_n = \frac{x_n^2}{10} + w_n, \quad (\text{observation model})$$

where  $y_n$  is a time series,  $x_n$  is the state vector. The system noise  $v_n$  and the observation noise  $w_n$  are assumed to be white noises which follow a Gaussian distribution and  $v_0 \sim N(0, 5)$ .

The algorithm of the particle filtering and smoothing are presented in Kitagawa (2020). For more details, please refer to Kitagawa (1996) and Doucet et al. (2001).

**Value**

An object of class "pfilter" which has a plot method. This is a list with the following components:

llkhood	log-likelihood.								
smooth.dist	marginal smoothed distribution of the trend $T(i, j)$ ( $i = 1, \dots, n, j = 1, \dots, 7$ ), where $n$ is the length of $y$ .								
	<table> <tr> <td>j = 4:</td> <td>50% point</td> </tr> <tr> <td>j = 3, 5:</td> <td>1-sigma points (15.87% and 84.14% points)</td> </tr> <tr> <td>j = 2, 6:</td> <td>2-sigma points (2.27% and 97.73% points)</td> </tr> <tr> <td>j = 1, 7:</td> <td>3-sigma points (0.13% and 99.87% points)</td> </tr> </table>	j = 4:	50% point	j = 3, 5:	1-sigma points (15.87% and 84.14% points)	j = 2, 6:	2-sigma points (2.27% and 97.73% points)	j = 1, 7:	3-sigma points (0.13% and 99.87% points)
j = 4:	50% point								
j = 3, 5:	1-sigma points (15.87% and 84.14% points)								
j = 2, 6:	2-sigma points (2.27% and 97.73% points)								
j = 1, 7:	3-sigma points (0.13% and 99.87% points)								

**References**

- Kitagawa, G. (1996) *Monte Carlo filter and smoother for non-Gaussian nonlinear state space models*, J. of Comp. and Graph. Statist., 5, 1-25.
- Doucet, A., de Freitas, N. and Gordon, N. (2001) *Sequential Monte Carlo Methods in Practice*,

Springer, New York.

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. Chapman & Hall/CRC.

### See Also

[pfilter](#) performs particle filtering and smoothing for linear non-Gaussian state-space model.

### Examples

```
data(NLmodel)
x <- NLmodel[, 2]
pfilterNL(x, m = 100000, lag = 20, sigma2 = 10.0, tau2 = 1.0,
          xrange = c(-20, 20), seed = 2019071117)
```

---

PfilterSample

*Sample Data for Particle Filter and Smoother*

---

### Description

An artificially generated sample data with shifting mean value.

### Usage

```
data(PfilterSample)
```

### Format

A time series of 400 observations.

### Details

This data generated by the following models;

$$y_n \sim N(\mu_n, 1), \quad \mu_n = \begin{cases} 0, & 1 \leq n \leq 100 \\ 1, & 101 \leq n \leq 200 \\ -1, & 201 \leq n \leq 300 \\ 0, & 301 \leq n \leq 400 \end{cases}$$

---

plot.boxcox

*Plot Box-Cox Transformed Data*

---

### Description

Plot original data and transformed data with minimum AIC.

**Usage**

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

**Arguments**

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

---

plot.lsqr	<i>Plot Fitted Trigonometric Polynomial</i>
-----------	---

---

**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	<i>Plot Smoothed Density Function</i>
-------------	---------------------------------------

---

**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" as returned 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 Posterior Distribution of Smoother*

---

**Description**

Plot posterior distribution (mean and standard deviations) of the smoother returned by [tsmooth](#).

**Usage**

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

**Arguments**

x                      an object of class "smooth".  
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, type = "v1", ...)
```

### Arguments

x                    an object of class "spg" as returned by [period](#) and [fftper](#).  
 type                type of plot. ("l": lines, "v1" : vertical lines)  
 ...                 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, tvv = NULL, dx = 2, dy = 0.25, ...)
```

**Arguments**

x	an object of class "tvspc".
tvv	time varying variance as returned by <a href="#">tvvar</a> .
dx	step width for the X axis.
dy	step width for the Y axis.
...	further graphical parameters may also be supplied as arguments.

**Examples**

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

z <- tvar(v$nordata, trend.order = 2, ar.order = 8, span = 20,
         outlier = c(630, 1026), tau2.ini = 6.6e-06, delta = 1.0e-06,
         plot = FALSE)

spec <- tvspc(z$arcoef, z$sigma2, span = 20, nf = 400)
plot(spec, tvv = v$tvv, dx = 2, dy = 0.10)
```

---

polreg

*Polynomial Regression Model*


---

**Description**

Estimate the trend using the AIC best polynomial regression model.

**Usage**

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

**Arguments**

<code>y</code>	a univariate time series.
<code>order</code>	maximum order of polynomial regression.
<code>plot</code>	logical. If TRUE (default), original data and trend component are plotted.
<code>...</code>	graphical arguments passed to <code>plot.polreg</code> .

**Value**

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

<code>order.maice</code>	MAICE (minimum AIC estimate) order.
<code>sigma2</code>	residual variance of the model with order $M$ . ( $0 \leq M \leq \text{order}$ )
<code>aic</code>	AIC of the model with order $M$ . ( $0 \leq M \leq \text{order}$ )
<code>daic</code>	AIC - minimum AIC.
<code>coef</code>	regression coefficients $A(I, M)$ with order $M$ . ( $1 \leq M \leq \text{order}$ , $1 \leq I \leq M$ )
<code>trend</code>	trend component.

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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)
```

---

Rainfall

*Rainfall Data*


---

**Description**

Number of rainy days in two years (1975-1976) at Tokyo, Japan.

**Usage**

```
data(Rainfall)
```



**Format**

Integer-valued time series of 366 observations.

**Source**

The data were obtained from Tokyo District Meteorological Observatory. <https://www.data.jma.go.jp/obd/stats/etrn/>

---

season	<i>Seasonal Adjustment</i>
--------	----------------------------

---

**Description**

Seasonal adjustment by state space modeling.

**Usage**

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

**Arguments**

y	a univariate time series with or without the tsp attribute.
trend.order	trend order (0, 1, 2 or 3).
seasonal.order	seasonal order (0, 1 or 2).
ar.order	AR order (0, 1, 2, 3, 4 or 5).
trade	logical; if TRUE, the model including trading day effect component is considered.
period	If the tsp attribute of y is NULL, valid number of seasons in one period in the case that seasonal.order > 0 and/or trade = TRUE.  4 : quarterly data 12 : monthly data 5 : daily data (5 days a week) 7 : daily data (7 days a week) 24 : hourly data
tau2.ini	initial estimate of variance of the system noise $\tau^2$ less than 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.

log.base	the letter "e" (default) or "10" specifying the base of logarithmic transformation. Valid only if log = TRUE.
minmax	lower and upper limits of observations.
plot	logical. If TRUE (default), trend, seasonal, AR and noise components are plotted.
...	graphical arguments passed to <code>plot.season</code> .

**Value**

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

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 = TRUE).
noise	noise component.
cov	covariance matrix of smoother.

**Note**

For time series with the tsp attribute, set frequency to period.

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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 data generated by simulation.
trend	initial values of trend component of length $m1$ , where $m1$ is trend order (1, 2). If NULL (default), trend order is 0.
seasonal.order	order of seasonal component model (0, 1, 2).
seasonal	if seasonal.order > 0, initial values of seasonal component of length $p - 1$ , where $p$ is period of one season.
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.
...	graphical arguments passed to <a href="#">plot.simulate</a> .

### Value

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

### References

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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)

n1 <- 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$tau2[3]
simssm(n = n1, trend, seasonal.order = m2, seasonal, arcoef, ar, tau1, tau2, tau3,
       sigma2 = z$sigma2, seed = 333)

```

---

Sunspot

*Sunspot 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	<i>Temperatures Data</i>
-------------	--------------------------

---

**Description**

The daily maximum temperatures in 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. <https://www.data.jma.go.jp/obd/stats/etrn/>

---

trend	<i>Trend Estimation</i>
-------	-------------------------

---

**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 $\tau^2$ . 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 component and residuals are plotted.
...	graphical arguments passed to <a href="#">plot.trend</a> .

**Details**

The trend model can be represented by a state space model

$$\begin{aligned}x_n &= Fx_{n-1} + Gv_n, \\ y_n &= Hx_n + w_n,\end{aligned}$$

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 components:

trend	trend component.
residual	residuals.
tau2	variance of the system noise $\tau^2$ .
sigma2	variance of the observational noise $\sigma^2$ .
llkhood	log-likelihood of the model.
aic	AIC.
cov	covariance matrix of smoother.

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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 $X(0   0)$ .
v0	initial state covariance matrix $V(0   0)$ .
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 vectors of the smoother and estimation error are plotted.
...	graphical arguments passed to <code>plot.smooth</code> .

**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", which is a list with 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. (2020) *Introduction to Time Series Modeling with Applications in R*. 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)
data(BLSALLFOOD)
BLS120 <- BLSALLFOOD[1:120]
z1 <- arfit(BLS120, plot = FALSE)
tau2 <- z1$sigma2

# m = maice.order, k=1
m1 <- z1$maice.order
arcoef <- z1$arcoef[[m1]]
f <- matrix(0.0e0, m1, m1)
f[1, ] <- arcoef
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)
z2 <- arfit(BLSALLFOOD, plot = FALSE)
tau2 <- z2$sigma2

# m = maice.order, k=1
m2 <- z2$maice.order
arcoef <- z2$arcoef[[m2]]
f <- matrix(0.0e0, m2, m2)
f[1, ] <- arcoef
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 $\tau^2$ . 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

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

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 [tvspc](#).

### Value

arcoef	time varying AR coefficients.
sigma2	variance of the observational noise $\sigma^2$ .
tau2	variance of the system noise $\tau^2$ .
llkhood	log-likelihood of the model.
aic	AIC.
parcor	PARCOR.

## References

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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.

## See Also

[tvspc](#), [plot.tvspc](#)

## 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)
```

---

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 power spectrum.

## Value

return an object of class "tvspc" giving power spectra, which has a plot method ([plot.tvspc](#)).

## References

- Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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 <- tvvar(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)
```

---

tvvar	<i>Time Varying Variance</i>
-------	------------------------------

---

## 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 $\tau^2$ . If tau2.ini = NULL, the most suitable value is chosen in $\tau^2 = 2^{-k}$ .
delta	search width.
plot	logical. If TRUE (default), transformed data, trend and residuals are plotted.
...	graphical arguments passed to the plot method.

## Details

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

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

where  $y_n$  is a Gaussian white noise with mean 0 and variance  $\sigma_n^2$ .  $s_m$  is distributed as a  $\chi^2$  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 has a plot method. This is a list with the following components:

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.
tname	the name of the univariate time series y.

## References

- Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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

<code>y</code>	a univariate time series.
<code>lag</code>	maximum lag. Default is $2\sqrt{n}$ , where $n$ is the length of the time series $y$ .
<code>minmax</code>	thresholds for outliers in low side and high side.
<code>plot</code>	logical. If TRUE (default), autocorrelations are plotted.
<code>...</code>	graphical arguments passed to the plot method.

## Value

An object of class "unicor" which has a plot method. This is a list with the following components:

<code>acov</code>	autocovariances.
<code>acor</code>	autocorrelations.
<code>acov.err</code>	error bound for autocovariances.
<code>acor.err</code>	error bound for autocorrelations.
<code>mean</code>	mean of $y$ .
<code>tname</code>	the name of the univariate time series $y$ .

**References**

Kitagawa, G. (2020) *Introduction to Time Series Modeling with Applications in R*. 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

## \* datasets

BLSALLFOOD, 8  
Haibara, 12  
HAKUSAN, 13  
MYE1F, 21  
Nikkei225, 25  
NLmodel, 26  
PfilterSample, 34  
Rainfall, 40  
Sunspot, 44  
Temperature, 45  
WHARD, 54

## \* package

TSSS-package, 3

## \* ts

arfit, 3  
armachar, 5  
armafit, 6  
armafit2, 7  
boxcox, 8  
crscor, 9  
fftper, 10  
klnfo, 14  
lsar, 15  
lsar.chgpt, 16  
lsqr, 17  
marfit, 18  
marlsq, 19  
marspc, 20  
ngsim, 21  
ngsmth, 23  
pdfunc, 26  
period, 28  
pfilter, 29  
pfilterNL, 32  
plot.boxcox, 34  
plot.lsqr, 35  
plot.ngsmth, 35  
plot.polreg, 36

plot.season, 36  
plot.simulate, 37  
plot.smooth, 37  
plot.spg, 38  
plot.trend, 38  
plot.tvspc, 39  
polreg, 39  
season, 41  
simssm, 43  
trend, 45  
tsmooth, 46  
tvar, 49  
tvspc, 50  
tvvar, 51  
unicor, 53

arfit, 3  
armachar, 5  
armafit, 6  
armafit2, 7  
  
BLSALLFOOD, 8  
boxcox, 8  
  
crscor, 9  
  
fftper, 10, 38  
  
Haibara, 12  
HAKUSAN, 13  
  
klnfo, 14  
  
lsar, 15  
lsar.chgpt, 16  
lsqr, 17, 35  
  
marfit, 18  
marlsq, 19  
marspc, 20  
MYE1F, 21

ngsim, 21, 37  
ngsmth, 23, 35  
Nikkei225, 25  
NLmodel, 26

pdfunc, 26  
period, 28, 38  
persp, 36  
pfilter, 29, 34  
pfilterNL, 32, 32  
PfilterSample, 34  
plot.boxcox, 9, 34  
plot.lsqr, 17, 35  
plot.ngsmth, 24, 35  
plot.pfilter (pfilter), 29  
plot.polreg, 36, 40  
plot.season, 36, 42  
plot.simulate, 22, 37, 43  
plot.smooth, 37, 47  
plot.spg, 11, 28, 38  
plot.trend, 38, 45  
plot.tvspc, 39, 50  
plot.tvvar (tvvar), 51  
polreg, 36, 39  
print.tvvar (tvvar), 51

Rainfall, 40

season, 36, 41  
simssm, 37, 43  
Sunspot, 44  
sunspot.year, 44

Temperature, 45  
trend, 38, 45  
tsmooth, 37, 46  
TSSS (TSSS-package), 3  
TSSS-package, 3  
tvar, 49  
tvspc, 39, 49, 50, 50  
tvvar, 39, 51

unicor, 53

WHARD, 54