Package 'LSTS'

October 1, 2015

Type Package

Title Locally Stationary Time Series

Version 1.0

Date 2015-09-29

Author Ricardo Olea <raolea@uc.cl>, Wilfredo Palma <wilfredo@mat.puc.cl>, Pilar Rubio <parubio@uc.cl>

Maintainer Ricardo Olea <raolea@uc.cl>

Imports grDevices, graphics, stats

Suggests rdatamarket

Description
Has a set of functions that allows stationary analysis and locally stationary time series analysis.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-01 23:54:34

R topics documented:

LSTS-package *LSTS Package*

Description

This package has a set of functions that allows stationary analysis and locally stationary time series analysis.

Details

Package:	LSTS
Type:	Package
Title:	Locally Stationary Time Series
Version:	1.0
Date:	2015-09-29
Functions:	`block.smooth.periodogram, Box.Ljung.Test, fdensity, hessian, LS.kalman, LS.summary, LS.whittle, LS.whittle.loglik, LS.whittle.loglik.sd, LS.whittle.theta, periodogram, smooth.periodogram, ts.diag`

Acknowledgment

The first author gratefully acknowledges the financial support from Fondecyt grant 11121128.

Author(s)

Ricardo Olea <raolea@uc.cl>, Wilfredo Palma <wilfredo@mat.puc.cl>, Pilar Rubio <parubio@uc.cl>

Maintainer: Ricardo Olea <raolea@uc.cl>

References

Brockwell, Peter J., and Richard A. Davis. *Introduction to time series and forecasting*. 2002. ISBN-13: 978-0387953519.

Dahlhaus, R. *Fitting time series models to nonstationary processes*. The Annals of Statistics. 1997; Vol. 25, No. 1:1-37.

Dahlhaus, R. and Giraitis, L. *On the optimal segment length for parameter estimates for locally stationary time series*. Journal of Time Series Analysis. 1998; 19(6):629-655.

Palma W. *Long-Memory Time Series. Theory and Methods*. 1st ed. New Jersey: John Wiley & Sons, Inc.; 2007. 285 p.

Olea R, Palma W. *An efficient estimator for locally stationary gaussian long-memory processes*. The Annals of Statistics. 2010; Vol. 38, No. 5:2958-2997.

Palma W, Olea R, & Ferreira G. *Estimation and forecasting of locally stationary processes*. Journal of Forecasting. 2013; Vol. 32, No. 1:86-96.

Examples

```
########################################
########## Tree Ring Aplication ##########
########################################

## Require: "rdatamarket"
library(rdatamarket)
malleco = dmlist("22tn")

y = malleco$Value
op = par(mfrow = c(1,2), cex = 0.9)
acf(y, lag = 10, ylim = c(-0.1,1), bty = "n", las = 1, main = "")
pacf(y, lag = 10, xlim = c(0,10), ylim = c(-0.1,1), bty = "n", las = 1, main = "")
par(op)
## AR(1) structure

y1 = y[1:250]
y2 = y[243:492]
y3 = y[485:734]
op = par(mfrow = c(2,3), cex = 0.9)
acf(y1, lag = 10, ylim = c(-0.1,1), bty = "n", las = 1, main = "block 1")
acf(y2, lag = 10, ylim = c(-0.1,1), bty = "n", las = 1, main = "block 2")
acf(y3, lag = 10, ylim = c(-0.1,1), bty = "n", las = 1, main = "block 3")
pacf(y1, lag = 10, xlim = c(0,10), ylim = c(-0.1,1), bty = "n", las = 1, main = "")
pacf(y2, lag = 10, xlim = c(0,10), ylim = c(-0.1,1), bty = "n", las = 1, main = "")
pacf(y3, lag = 10, xlim = c(0,10), ylim = c(-0.1,1), bty = "n", las = 1, main = "")
par(op)

block.smooth.periodogram(y, spar.freq = 0.8, spar.time = 0.8, theta = 90, phi = 0)

# Analysis by blocks of phi and sigma parameters
T. = length(y)
N = 200
S = 100
M = trunc((T. - N)/S + 1)
table = c()
for(j in 1:M){
x = y[(1 + S*(j-1)):(N + S*(j-1))]
table = rbind(table,nlminb(start = c(0.65, 0.15), N = N, objective = LS.whittle.loglik,
                           series = x, order = c(p = 1, q = 0))$par)
}
u = (N/2 + S*(1:M-1))/T.
table = as.data.frame(cbind(u, table))
colnames(table) = c("u", "phi", "sigma")

op = par(mfrow = c(1,2), cex = 0.8)
spar = 0.6
plot(smooth.spline(phi, spar = spar)$y ~ u, data = table, pch = 20, ylim = c(0,1),
     xlim = c(0,1), las = 1, bty = "n", ylab = expression(phi(u)))
```

```
plot(smooth.spline(sigma, spar = spar)$y ~ u, data = table, pch = 20, ylim = c(0,0.2),
     xlim = c(0,1), las = 1, bty = "n", ylab = expression(phi(u)))
par(op)

## start parameters
phi = smooth.spline(table$phi, spar = spar)$y
fit.1 = nls(phi ~ a0+a1*u, start = list(a0 = 0.65, a1 = 0.00))
sigma = smooth.spline(table$sigma, spar = spar)$y
fit.2 = nls(sigma ~ b0+b1*u, start = list(b0 = 0.65, b1 = 0.00))

fit = LS.whittle(series = y, start = c(coef(fit.1), coef(fit.2)), order = c(p = 1, q = 0),
                 ar.order = c(1), sd.order = 1, N = 180, n.ahead = 10)

## Summary
LS.summary(fit)

## Diagnostic
ts.diag(fit$residuals)

## Fitted Values
op = par(mfrow = c(1,2), cex = 0.8)
spar = 0.6
plot(smooth.spline(phi, spar = spar)$y ~ u, data = table, pch = 20, ylim = c(0,1),
     xlim = c(0,1), las = 1, bty = "n", ylab = expression(phi(u)))
lines(fit$coef[1]+fit$coef[2]*u~u)
plot(smooth.spline(sigma, spar = spar)$y ~ u, data = table, pch = 20, ylim = c(0,0.2),
     xlim = c(0,1), las = 1, bty = "n", ylab = expression(sigma(u)))
lines(fit$coef[3]+fit$coef[4]*u~u)
par(op)

plot(fit$series, type = "l")
lines(fit$fitted.values, col = "red", lty = 1)
for(alpha in seq(0.05,1.00, 0.01)){
lines(fit$pred + fit$se * qnorm(1-alpha/2), col = gray(1-alpha))
lines(fit$pred - fit$se * qnorm(1-alpha/2), col = gray(1-alpha))
}
```

`block.smooth.periodogram` *Smooth Periodogram by blocks*

Description

Plot in 3D the smoothing periodogram of a time series, by blocks or windows.

Usage

```
block.smooth.periodogram(y, x = NULL, N = NULL, S = NULL, p = 0.25,
                         spar.freq = 0, spar.time = 0, theta = 0, phi = 0,
                         xlim = NULL, ylim = NULL, zlim = NULL, ylab = "Time",
                         palette.col = NULL)
```

Arguments

y	data vector.
x	optional vector, if x=NULL then the function uses x=1:n where n is the length of y. More details in 'y' argument from persp function.
N	value corresponding to the length of the window to compute periodogram. If N=NULL then the function will use $N = \text{trunc}(n^{0.8})$, see Dahlhaus (1998) where n is the length of the y vector.
S	value corresponding to the lag with which will be taking the blocks or windows to calculate the periodogram.
p	value used if it is desired that S is proportional to N. By default p=0.25, if S and N are not entered.
spar.freq, spar.time	smoothing parameter, typically (but not necessarily) in (0,1].
theta, phi	angles defining the viewing direction. theta gives the azimuthal direction and phi the colatitude.
xlim, ylim, zlim	x-,y- and z-limits. They are NULL by default and they are optionals parameters.
ylab	title for 'y' axis. Optional argument, by default is ylab="Time". This must be character strings.
palette.col	colors palette.

Details

The number of windows of the function is $M = \text{trunc}((n - N)/S + 1)$, where trunc truncates de entered value and *n* is the length of the vector y. All windows are of the same length N, if this value isn't entered by user then is computed as $N = \text{trunc}(n^{0.8})$ (Dahlhaus).

block.smooth.periodogram computes the periodogram in each of the *M* windows and then smoothes it two times with smooth.spline function; the first time using spar.freq parameter and the second time with spar.time. These windows overlap between them.

The surface is then viewed by looking at the origin from a direction defined by theta and phi. If theta and phi are both zero the viewing direction is directly down the negative y axis. Changing theta will vary the azimuth and changing phi the colatitude.

Author(s)

Ricardo Olea <raolea@uc.cl>

References

Dahlhaus, R. *Fitting time series models to nonstationary processes*. The Annals of Statistics. 1997; Vol. 25, No. 1:1-37.

Dahlhaus, R. and Giraitis, L. *On the optimal segment length for parameter estimates for locally stationary time series*. Journal of Time Series Analysis. 1998; 19(6):629-655.

See Also

See periodogram and persp to more details.

Examples

```
## Require "rdatamarket"
library(rdatamarket)

malleco = dmlist("22tn")
mammothcreek = dmlist("22r7")


## Example 1
block.smooth.periodogram(y = malleco$Value, x = malleco$Year, spar.freq = .7,
                         spar.time = .7, theta = 30, phi =  0, N = 300, S = 50,
                         ylim = c(1200,2000), ylab = "Year")
block.smooth.periodogram(y = malleco$Value, x = malleco$Year, spar.freq = .7,
                         spar.time = .7, theta = 45, phi = 45, N = 300, S = 50,
                         ylim = c(1200,2000), ylab = "Year")
block.smooth.periodogram(y = malleco$Value, x = malleco$Year, spar.freq = .7,
                         spar.time = .7, theta = 90, phi =  0, N = 300, S = 50,
                         ylim = c(1200,2000), ylab = "Year")

## Example 2
block.smooth.periodogram(y = mammothcreek$Value, x = mammothcreek$Year, spar.freq = .7,
                         spar.time = .7, theta = 30, phi =  0, N = 400, S = 50,
                         ylim = c(-10,2000), ylab = "Year")
block.smooth.periodogram(y = mammothcreek$Value, x = mammothcreek$Year, spar.freq = .7,
                         spar.time = .7, theta = 45, phi = 45, N = 400, S = 50,
                         ylim = c(-10,2000), ylab = "Year")
block.smooth.periodogram(y = mammothcreek$Value, x = mammothcreek$Year, spar.freq = .7,
                         spar.time = .7, theta = 90, phi = 0, N = 400, S = 50,
                         ylim = c(-10,2000), ylab = "Year")

## Example 3: Simulated AR(2)
set.seed(2015)
ts.sim = arima.sim(n = 1000, model = list(order = c(2,0,0), ar = c(1.3, -0.6)))
block.smooth.periodogram(y = ts.sim, spar.freq = .9, spar.time = .9, theta = 30, phi = 00,
                         N = 500, S = 100, ylab = "Time")
block.smooth.periodogram(y = ts.sim, spar.freq = .9, spar.time = .9, theta = 00, phi = 00,
                         N = 500, S = 100, ylab = "Time")
block.smooth.periodogram(y = ts.sim, spar.freq = .9, spar.time = .9, theta = 90, phi = 00,
                         N = 500, S = 100, ylab = "Time")
block.smooth.periodogram(y = ts.sim, spar.freq = .9, spar.time = .9, theta = 45, phi = 15,
                         N = 500, S = 100, ylab = "Time")
block.smooth.periodogram(y = ts.sim, spar.freq = .9, spar.time = .9, theta = 45, phi = 15,
                         N = 500, S = 100, ylab = "Time",
                         palette.col = gray(level = seq(0.2,0.9,0.1 )))
```

`Box.Ljung.Test` *Ljung-Box Test Plot*

Description

Returns plot of the p-values Ljung-Box test.

Usage

```
Box.Ljung.Test(z, lag = NULL, main = NULL)
```

Arguments

`z`	numeric vector or univariate time series.
`lag`	lag or `lag.max` used in the `acf` of the time serie. If `lag=NULL` then the function will use `lag=10`.
`main`	an overall title for the plot. Optional argument, by default will print *p values for Ljung-Box statistic*.

Details

The Ljung-Box test is used to check if exists autocorrelation in a time series. The statistic is

$$Q = n(n+2) \cdot \sum_{j=1}^{h} \hat{\rho}(j)^2/(n-j)$$

with *n* the number of observations and $\hat{\rho}(j)$ the autocorrelation coefficient in the sample when the lag is *j*. `Box.Ljung.Test` computes Q and returns the p-values graph with lag *j*.

Author(s)

Ricardo Olea <raolea@uc.cl>

References

Brockwell, Peter J., and Richard A. Davis. *Introduction to time series and forecasting*. 2002. ISBN-13: 978-0387953519.

Ljung, G.M. and Box, G.E.P. (1978), *On a measure of lack of fit in time series models*, Biometrika, 65, 297-303.

Examples

```
z = rnorm(500)
Box.Ljung.Test(z, lag=15)
ts.diag(z)
```

`fdensity` *Spectral density*

Description

Returns theoretical spectral density evaluated in ARMA and ARFIMA processes.

Usage

```
fdensity(ar = numeric(), ma = numeric(), d = 0, sd = 1, lambda = NULL)
```

Arguments

`ar, ma`	AR or MA vector, respectively. If the time serie doesn't have AR (or MA) term then omit it. To more details see examples.
`d`	d is a long-memory parameter. If d is zero, then the process is ARMA(p,q)
`sd`	Noise scale factor, by default is 1.
`lambda`	λ parameter on which the spectral density is calculated/computed. If `lambda=NULL` then it is considered a sequence between 0 and π.

Details

The spectral density of an ARFIMA(p,d,q) processes is

$$f(\lambda) = \frac{\sigma^2}{2\pi} \cdot \Big(2\sin(\lambda/2)\Big)^{-2d} \cdot \frac{\Big|\theta\Big(\exp\Big(-i\lambda\Big)\Big)\Big|^2}{\Big|\phi\Big(\exp\Big(-i\lambda\Big)\Big)\Big|^2}$$

with $-\pi \leq \lambda \leq \pi$ and $-1 < d < 1/2$. $|x|$ is the Mod of x. `fdensity` returns the values corresponding to $f(\lambda)$. When d is zero, the spectral density corresponds to an ARMA(p,q).

Author(s)

Ricardo Olea <raolea@uc.cl>

References

Brockwell, Peter J., and Richard A. Davis. *Introduction to time series and forecasting*. 2002. ISBN-13: 978-0387953519.

Palma W. *Long-Memory Time Series. Theory and Methods*. 1st ed. New Jersey: John Wiley & Sons, Inc.; 2007. 285 p.

Examples

```
## Example 1: Spectral Density AR(1)
lambda = seq(0,pi,0.01)
f = fdensity(ar = 0.5, lambda = lambda)
plot(f~lambda, bty = "n", type = "l", las = 1, xlab = expression("Frequency"),
     ylab = expression("Spectral Density"))


## Example 2: Spectral Density AR(2)
lambda = seq(0,pi,0.01)
f = fdensity(ar = c(1.3,-0.6), lambda = lambda, sd=10)
plot(f~lambda, bty = "n", type = "l", las = 1, xlab = expression("Frequency"),
     ylab = expression("Spectral Density"))


## Spectral Density ARMA(1,1)
lambda = seq(0,pi,0.01)
f = fdensity(ar = 0.5, ma = 0.8, lambda = lambda)
plot(f~lambda, bty = "n", type = "l", las = 1, xlab = expression("Frequency"),
     ylab = expression("Spectral Density"))


## Spectral Density ARFIMA(1,d,1)
lambda = seq(0,pi,0.01)
f = fdensity(ar = 0.5, ma = 0.8, d = 0.2, lambda = lambda)
plot(f~lambda, bty = "n", type = "l", las = 1, xlab = expression("Frequency"),
     ylab = expression("Spectral Density"))
```

---

hessian *Hessian Matrix*

Description

Computes numerical approximation to Hessian of `f`, evaluated at `x0`. Usually need to pass additional parameters (e.g. data). N.B. this uses no numerical sophistication.

Usage

```
hessian(f, x0, ...)
```

Arguments

`f`	name of function that defines log likelihood (or negative of it).
`x0`	scalar or vector of parameters that give the point at which you want the hessian estimated (usually will be the mle).
`...`	additional arguments passed to `f`.

Value

An `n x n` matrix of 2nd derivatives, where *n* is the length of `x0`.

Author(s)

Wilfredo Palma <wilfredo@mat.puc.cl>

Examples

```
## Ex 1:  Variance of the maximum likelihood estimator for mu parameter in gaussian data
loglik = function(series, x, sd = 1){
-sum(log(dnorm(series, mean = x, sd = sd)))
}
n = 500
series = rnorm(500, mean = 10, sd = 2)
sqrt(c(var(series)/n,diag(solve(hessian(f = loglik, x = mean(series), series = series,
                          sd = sd(series))))))


## Ex 2:  Variance of the maximum likelihood estimator for phi parameter AR(1)
## in gaussian data
loglik = function(series, x, sd = 1){
n = length(series)
-(sum(log(dnorm(series[2:n], mean = x * series[1:(n-1)], sd = sd))) +
log(dnorm(series[1], mean = 0, sd = sqrt(sd^2/(1-x^2)))))
}
n = 1000
series = arima.sim(n, model = list(c(1,0,0), ar = 0.7))
fit = arima(series, c(1,0,0), include.mean = FALSE)
round(c(fit$var.coef,diag(solve(hessian(f = loglik, x = fit$coef, series = series,
      sd = sqrt(fit$sigma2))))),6)


## Ex 3:  Variance of the whittle maximum likelihood estimator for phi parameter
## AR(1) in gaussian  data
loglik = function(series, x, sd = 1){
n = length(series)
aux = periodogram(series, plot = FALSE)
lambda = aux$lambda
I = aux$periodogram
f = fdensity(ar = x, sd = sd, lambda = lambda)
lik = sum(log(f)+I/f)
lik = lik/n
}
n = 500
series = arima.sim(n, model = list(c(1,0,0), ar = 0.7))
fit = arima(series, c(1,0,0), include.mean = FALSE)
round(c(fit$var.coef,diag(solve(hessian(f = loglik, x = fit$coef, series = series,
      sd = sqrt(fit$sigma2))))/n),4)
```

LS.kalman *Kalman filter for locally stationary processes*

Description

This function run the state-space equations for expansion infinite of moving average in processes LS-ARMA or LS-ARFIMA.

Usage

```
LS.kalman(series, start, order = c(p = 0, q = 0), ar.order = NULL, ma.order = NULL,
  sd.order = NULL, d.order = NULL, include.d = FALSE, m = NULL)
```

Arguments

series	univariate time series.
start	numeric vector, initial values for parameters to run the model.
order	vector corresponding to ARMA model entered.
ar.order, ma.order	AR and MA polimonial order respectively.
sd.order	polinomial order noise scale factor.
d.order	d polinomial order, where d is the ARFIMA parameter.
include.d	logical argument for ARFIMA models. If include.d=FALSE then the model is an ARMA process.
m	truncation order of the MA infinity process. By default $m = 0.25 * n^{0.8}$ where n the length of series.

Details

The model fit is done using the Whittle likelihood, while the generation of innovations is through Kalman Filter. Details about ar.order, ma.order, sd.order and d.order can be viewed in LS.whittle.loglik.

Value

A list with:

residuals	standard residuals.
fitted.values	model fitted values.
delta	variance prediction error.

Author(s)

Ricardo Olea <raolea@uc.cl>

References

Brockwell, Peter J., and Richard A. Davis. *Introduction to time series and forecasting*. 2002. ISBN-13: 978-0387953519.

Palma W. *Long-Memory Time Series. Theory and Methods*. 1st ed. New Jersey: John Wiley & Sons, Inc.; 2007. 285 p.

Palma W, Olea R, & Ferreira G. *Estimation and forecasting of locally stationary processes*. Journal of Forecasting. 2013; VOl. 32, No. 1:86-96.

`LS.summary` *Summary for Locally Stationary Time Series*

Description

Function used to produce summaries of the results to Whittle estimator to Locally Stationary Time Series (`LS.whittle` function).

Usage

```
LS.summary(object)
```

Arguments

`object`	`LS.whittle` function.

Value

A list with the following components:

`summary`	a resume table with estimate, std. error, z-value and p-value of the model.
`aic`	AIC of the model.
`npar`	number of parameters in the model.

Author(s)

Ricardo Olea <raolea@uc.cl>

LS.whittle *Whittle estimator to Locally Stationary Time Series*

Description

This function computes Whittle estimator to LS-ARMA and LS-ARFIMA models.

Usage

```
LS.whittle(series, start, order = c(p = 0, q = 0), ar.order = NULL,
           ma.order = NULL, sd.order = NULL, d.order = NULL, include.d = FALSE,
           N = NULL, S = NULL, include.taper = TRUE, control = list(),
           lower = -Inf, upper = Inf, m = NULL, n.ahead = 0)
```

Arguments

series	data vector.
start	numeric vector, initial values for the parameters to be optimized.
order	vector with the specification of the ARMA model: the two integers components (p, q) are the AR order and the MA order.
ar.order, ma.order	AR and MA polimonial order, respectively. See details below.
sd.order	polinomial order noise scale factor. See details below.
d.order	d polinomial order, where d is the ARFIMA parameter.
include.d	logical argument for ARFIMA models. If include.d=FALSE then the model is an ARMA process.
N	value corresponding to the length of the window to compute periodogram. If N=NULL then the function will use $N = \text{trunc}(n^{0.8})$, see Dahlhaus (1998) where n is the length of the y vector.
S	value corresponding to the lag with which will go taking the blocks or windows.
include.taper	logical argument that by default is TRUE. See periodogram.
control	A list of control parameters. More details in nlminb .
lower, upper	vectors of lower and upper bounds, replicated to be as long as start. If unspecified, all parameters are assumed to be unconstrained.
m	truncation order of the MA infinity process, by default $m = 0.25 * n^{0.8}$. Parameter used in LS.kalman.
n.ahead	The number of steps ahead for which prediction is required. By default is zero.

Details

This function estimates the parameters in models: LS-ARMA

$$\Phi(t/T,\, B)\, Y_{t,T} = \Theta(t/T,\, B)\, \sigma(t/T)\, \varepsilon_t$$

and LS-ARFIMA

$$\Phi(t/T,\, B)\, Y_{t,T} = \Theta(t/T,\, B)\, (1-B)^{-d(t/T)}\, \sigma(t/T)\, \varepsilon_t,$$

with infinite moving average expansion

$$Y_{t,T} = \sigma(t/T) \sum_{j=0}^{\infty} \psi(t/T)\, \varepsilon_t,$$

for $t = 1, \ldots, T$, where for $u = t/T \in [0,1]$, $\Phi(u,B) = 1 + \phi_1(u)B + \cdots + \phi_p(u)B^p$ is an autoregressive polynomial, $\Theta(u,B) = 1+\theta_1(u)B+\cdots+\theta_q(u)B^q$ is a moving average polynomial, $d(u)$ is a long-memory parameter, $\sigma(u)$ is a noise scale factor and $\{\varepsilon_t\}$ is a Gaussian white noise sequence with zero mean and unit variance. This class of models extends the well-known ARMA and ARFIMA process, which is obtained when the components $\Phi(u,B)$, $\Theta(u,B)$, $d(u)$ and $\sigma(u)$ do not depend on u.

The evolution of these models can be specified in terms of a general class of functions. For example, let $\{g_j(u)\}$, $j = 1, 2, \ldots$, be a basis for a space of smoothly varying functions and let $d_\theta(u)$ be the time-varying long-memory parameter in model LS-ARFIMA. Then we could write $d_\theta(u)$ in terms of the basis $\{g_j(u) = u^j\}$ as follows $d_\theta(u) = \sum_{j=0}^{k} \alpha_j\, g_j(u)$ for unknown values of k and $\theta = (\alpha_0,\, \alpha_1,\, \ldots,\, \alpha_k)'$. In this situation, estimating θ involves determining k and estimating the coefficients $\alpha_0,\, \alpha_1,\, \ldots,\, \alpha_k$.

`LS.whittle` optimizes `LS.whittle.loglik` as objective function using `nlminb` function, for both LS-ARMA (`include.d=FALSE`) and LS-ARFIMA (`include.d=TRUE`) models. Also computes Kalman filter with `LS.kalman` and this values are given in `var.coef` in the output.

Value

A list with the following components:

`coef`	The best set of parameters found.
`var.coef`	covariance matrix approximated for maximum likelihood estimator $\hat{\theta}$ of $\theta := (\theta_1, \ldots, \theta_k)'$. This matrix is approximated by H^{-1}/n, where H is the Hessian matrix $[\partial^2 \ell(\theta)/\partial\theta_i \partial\theta_j]_{i,j=1}^{k}$.
`loglik`	log-likelihood of `coef`, calculated with `LS.whittle.loglik`.
`aic`	Akaike's 'An Information Criterion', for one fitted model LS-ARMA or LS-ARFIMA. The formula is -2*log-likelihood + 2*npar/n, where *npar* represents the number of parameters in the fitted model and *n* equal to the length of the `series`.
`series`	original time serie.
`residuals`	standard residuals.
`fitted.values`	model fitted values.
`pred`	predictions of the model.
`se`	the estimated standard errors.
`model`	A list representing the fitted model.

Author(s)

Ricardo Olea <raolea@uc.cl>

See Also

nlminb for optimization. LS.kalman for Kalman filter.

Examples

```
## Require "rdatamarket"
library(rdatamarket)
malleco = dmlist("22tn")
y = malleco$Value

# Analysis by blocks of phi and sigma parameters
T. = length(y)
N = 200
S = 100
M = trunc((T. - N)/S + 1)
table = c()
for(j in 1:M){
x = y[(1 + S*(j-1)):(N + S*(j-1))]
table = rbind(table,nlminb(start = c(0.65, 0.15), N = N, objective = LS.whittle.loglik,
                           series = x, order = c(p = 1, q = 0))$par)
}
u = (N/2 + S*(1:M-1))/T.
table = as.data.frame(cbind(u, table))
colnames(table) = c("u", "phi", "sigma")

spar = 0.6
## start parameters
phi = smooth.spline(table$phi, spar = spar)$y
fit.1 = nls(phi ~ a0+a1*u, start = list(a0 = 0.65, a1 = 0.00))
sigma = smooth.spline(table$sigma, spar = spar)$y
fit.2 = nls(sigma ~ b0+b1*u, start = list(b0 = 0.65, b1 = 0.00))

fit = LS.whittle(series = y, start = c(coef(fit.1), coef(fit.2)), order = c(p = 1, q = 0),
                 ar.order = c(1), sd.order = 1, N = 180, n.ahead = 10)

names(fit)
fit$coef
fit$loglik
```

LS.whittle.loglik *Locally Stationary Whittle log-likelihood Function*

Description

This function computes Whittle estimator for LS-ARMA and LS-ARFIMA models, in data with mean zero. If mean is not zero, then it is subtracted to data.

Usage

```
LS.whittle.loglik(x, series, order = c(p = 0, q = 0), ar.order = NULL,
                  ma.order = NULL, sd.order = NULL, d.order = NULL,
                  include.d = FALSE, N = NULL, S = NULL, include.taper = TRUE)
```

Arguments

x	parameter vector.
series	univariate time series.
order	vector with the specification of the ARMA model: the two integer components (p, q) are the AR order and the MA order.
ar.order, ma.order	AR and MA polimonial order, respectively. See details below.
sd.order	polinomial order noise scale factor.
d.order	d polinomial order, where d is the ARFIMA parameter.
include.d	logical argument for ARFIMA models. If include.d=FALSE then the model is an ARMA process.
N	value corresponding to the length of the window to compute periodogram. If N=NULL then the function will use $N = \text{trunc}(n^{0.8})$, see Dahlhaus (1998) where n is the length of the y vector.
S	value corresponding to the lag with which will go taking the blocks or windows.
include.taper	logical argument that by default is TRUE. See periodogram.

Details

The estimation of the time-varying parameters can be carried out by means of the Whittle log-likelihood function proposed by Dahlhaus (1997),

$$L_n(\theta) = \frac{1}{4\pi}\frac{1}{M}\int_{-\pi}^{\pi}\left\{log f_\theta(u_j,\lambda) + \frac{I_N(u_j,\lambda)}{f_\theta(u_j,\lambda)}\right\} d\lambda$$

where M is the number of blocks, N the length of the series per block, $n = S(M-1)+N$, S is the shift from block to block, $u_j = t_j/n$, $t_j = S(j-1)+N/2$, $j = 1,\ldots,M$ and λ the Fourier frequencies in the block ($2\,\pi\,k/N$, $k = 1,\ldots,N$).

Author(s)

Ricardo Olea <raolea@uc.cl>

References

Dahlhaus, R. *Fitting time series models to nonstationary processes*. The Annals of Statistics. 1997; Vol. 25, No. 1:1-37.

Olea R, Palma W. *An efficient estimator for locally stationary gaussian long-memory processes*. The Annals of Statistics. 2010; Vol. 38, No. 5:2958-2997.

`LS.whittle.loglik.sd` *Locally Stationary Whittle Log-likelihood sigma*

Description

This function calculates log-likelihood with known θ, through `LS.whittle.loglik` function.

Usage

```
LS.whittle.loglik.sd(x, series, order = c(p = 0, q = 0), ar.order = NULL,
                     ma.order = NULL, sd.order = NULL, d.order = NULL,
                    include.d = FALSE, N = NULL, S = NULL, include.taper = TRUE,
                     theta.par = numeric())
```

Arguments

`x`	parameter vector.
`series`	univariate time series.
`order`	vector with the specification of the ARMA model: the two integer components (p, q) are the AR order and the MA order.
`ar.order, ma.order`	AR and MA polimonial order, respectively.
`sd.order`	polinomial order noise scale factor.
`d.order`	d polinomial order, where d is the `ARFIMA` parameter.
`include.d`	logical argument for `ARFIMA` models. If `include.d=FALSE` then the model is an ARMA process.
`N`	value corresponding to the length of the window to compute periodogram. If `N=NULL` then the function will use $N = \text{trunc}(n^{0.8})$, see Dahlhaus (1998) where n is the length of the `y` vector.
`S`	value corresponding to the lag with which will go taking the blocks or windows.
`include.taper`	logical argument that by default is `TRUE`. See `periodogram`.
`theta.par`	vector with the known parameters of the model.

Details

This function computes `LS.whittle.loglik` with `x` as `x = c(theta.par, x)`.

Author(s)

Ricardo Olea <raolea@uc.cl>

LS.whittle.loglik.theta

Locally Stationary Whittle Log-likelihood theta

Description

Calculate the log-likelihood with σ known, through LS.whittle.loglik function.

Usage

```
LS.whittle.loglik.theta(x, series, order = c(p = 0, q = 0), ar.order = NULL,
                        ma.order = NULL, sd.order = NULL, d.order = NULL,
                        include.d = FALSE, N = NULL, S = NULL,
                        include.taper = TRUE, sd.par = 1)
```

Arguments

x	parameter vector.
series	univariate time series.
order	vector with the specification of the ARMA model: the two integer components (p, q) are the AR order and the MA order.
ar.order, ma.order	AR and MA polimonial order respectively.
sd.order	polinomial order noise scale factor.
d.order	d polinomial order, where d is the ARFIMA parameter.
include.d	logical argument for ARFIMA models. If include.d=FALSE then the model is an ARMA process.
N	value corresponding to the length of the window to compute periodogram. If N=NULL then the function will use $N = \text{trunc}(n^{0.8})$, see Dahlhaus (1998) where n is the length of the y vector.
S	value corresponding to the lag with which will go taking the blocks or windows.
include.taper	logical argument that by default is TRUE. See periodogram.
sd.par	value corresponding to known variance.

Details

This function computes LS.whittle.loglik with x as x = c(x, sd.par).

Author(s)

Ricardo Olea <raolea@uc.cl>

`periodogram` *Periodogram function*

Description

This function computes the periodogram from a stationary time serie. Returns the periodogram, its graph and the Fourier frequency.

Usage

```
periodogram(y, plot = TRUE, include.taper = FALSE)
```

Arguments

`y`	data vector.
`plot`	logical argument which allows to plot the periodogram.
`include.taper`	logical argument which by default is `FALSE`. If `include.taper=TRUE` then `y` is multiplied by $0.5 * (1 - cos(2\pi(n-1)/n))$(*cosine bell*).

Details

The tapered periodogram it is given by

$$I(\lambda) = \frac{|D_n(\lambda)|^2}{2\pi H_{2,n}(0)}$$

with $D(\lambda) = \sum_{s=0}^{n-1} h\left(\frac{s}{N}\right) y_{s+1}\, e^{-i\,\lambda\, s}$, $H_{k,n} = \sum_{s=0}^{n-1} h\left(\frac{s}{N}\right)^k\, e^{-i\,\lambda\, s}$ and λ are Fourier frequencies defined as $2\pi k/n$, with $k = 1, \ldots, n$.\

The data taper used is the cosine bell function, $h(x) = \frac{1}{2}[1 - cos(2\pi x)]$. If the series has missing data, these are replaced by the average of the data and n it is corrected by \$n-N\$, where N is the amount of missing values of serie.

The plot of the periodogram is `periodogram` values vs. λ.

Value

`periodogram` is a vector with values of the periodogram of the serie, while `lambda` is a vector with values corresponding to Fourier frequency. The graph is `periodogram ~ lambda`.

Author(s)

Ricardo Olea <raolea@uc.cl>

References

Brockwell, Peter J., and Richard A. Davis. *Introduction to time series and forecasting*. 2002. ISBN-13: 978-0387953519.

Dahlhaus, R. *Fitting time series models to nonstationary processes*. The Annals of Statistics. 1997; Vol. 25, No. 1:1-37.

See Also

fft, Mod to 'periodogram' definition and smooth.periodogram in this package.

Examples

```
## Require "rdatamarket"
library(rdatamarket)

## Database
malleco = dmlist("22tn")
mammothcreek = dmlist("22r7")

periodogram(malleco$Value)
periodogram(mammothcreek$Value)
```

---

smooth.periodogram *Smoothing periodogram*

Description

This function returns the smoothing periodogram of a stationary time serie, its plot and its Fourier frequency.

Usage

```
smooth.periodogram(y, plot = TRUE, spar = 0)
```

Arguments

y	data vector.
plot	logical argument which allows to plot the periodogram.
spar	smoothing parameter, typically (but not necessarily) in (0,1].

Details

smooth.periodogram computes the periodogram from y vector and then smooth it with *smoothing spline* method, which basically approximates a curve using a cubic spline (see more details in smooth.spline). λ is the Fourier frequency obtained through periodogram.

It must have caution with the minimum length of y, because smooth.spline requires the entered vector has at least length 4 and the length of y does not equal to the length of the data of the periodogram that smooth.spline receives. If it presents problems with tol (**tol**erance), see smooth.spline.

Value

A list with the following components:

`smooth.periodogram` vector with the smoothing periodogram values.

`lambda` vector coresponding to Fourier frequency. See details above.

Author(s)

Ricardo Olea <raolea@uc.cl>

See Also

To more information please check `smooth.spline` and `periodogram`.

Examples

```
## Require "rdatamarket"
library(rdatamarket)

malleco = dmlist("22tn")
mammothcreek = dmlist("22r7")

## Example 1: malleco
periodogram(malleco$Value)
aux = smooth.periodogram(malleco$Value, plot = FALSE, spar = .8)
lines(aux$smooth.periodogram ~ aux$lambda, lwd = 2, col = "orange")

## Example 2: mammothcreek$Value
periodogram(mammothcreek$Value)
aux = smooth.periodogram(mammothcreek$Value, plot = FALSE, spar = .4)
lines(aux$smooth.periodogram ~ aux$lambda, lwd = 2, col = "orange")

## Example 3: AR(1) simulated
ts.sim = arima.sim(n = 1000, model = list(order = c(1,0,0), ar = 0.7))
periodogram(ts.sim)
aux = smooth.periodogram(ts.sim, plot = FALSE, spar = .7)
lines(aux$smooth.periodogram ~ aux$lambda, lwd = 2, col = "orange")
lines(fdensity(ar = 0.7, lambda = aux$lambda)~aux$lambda, col = "red")
```

---

ts.diag *Diagnostic Plots for Time Series fits*

Description

Plot time-series diagnostics.

Usage

```
ts.diag(x, lag = 10, cex = 0.5)
```

Arguments

x	residuals of the fitted time series model.
lag	maximum lag at which to calculate the acf and Ljung-Box test.
cex	optional argument of par.

Details

This function plot the residuals, the autocorrelation function of the residuals (ACF) and the p-values of the Ljung-Box Test for all lags up to lag.

Value

Diagnostics are plotted.

Author(s)

Ricardo Olea <raolea@uc.cl>

Examples

```
z = rnorm(500)
Box.Ljung.Test(z, lag=15)
ts.diag(z)
```

Index