# Package 'LSTS'

May 7, 2026

**Type** Package

**Title** Locally Stationary Time Series

**Version** 2.1

**Description** A set of functions that allow stationary analysis and locally stationary time series analysis.

**URL** https://pacha.dev/LSTS/

**BugReports** https://github.com/pachadotdev/LSTS/issues/

**Imports** stats, Rdpack, ggplot2, scales, patchwork

**RdMacros** Rdpack

**Depends** R (>= 3.6.0)

**License** Apache License (>= 2)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**Suggests** testthat (>= 2.1.0)

**NeedsCompilation** no

**Author** Ricardo Olea [aut, cph],
Wilfredo Palma [aut, cph],
Pilar Rubio [aut],
Mauricio Vargas [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1017-7574>)

**Maintainer** Mauricio Vargas <mavargas11@uc.cl>

**Repository** CRAN

**Date/Publication** 2021-07-29 16:00:02 UTC

## Contents

---

`block.smooth.periodogram`

*Smooth Periodogram by Blocks*

---

## Description

Plots the contour plot of 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
)
```

## Arguments

| | |
|---|---|
| `y` | (type: numeric) data vector |
| `x` | (type: numeric) optional vector, if `x = NULL` then the function uses $(1, \ldots, n)$ where `n` is the length of `y`. |
| `N` | (type: numeric) 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 and Giraitis (1998) where $n$ is the length of the `y` vector. |
| `S` | (type: numeric) value corresponding to the lag with which will be taking the blocks or windows to calculate the periodogram. |
| `p` | (type: numeric) 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` | (type: numeric) smoothing parameter, typically (but not necessarily) in $(0, 1]$. |
| `spar.time` | (type: numeric) smoothing parameter, typically (but not necessarily) in $(0, 1]$. |

### 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). `LSTS_spb` 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.

### Value

A ggplot object.

### References

For more information on theoretical foundations and estimation methods see Dahlhaus R, others (1997). "Fitting time series models to nonstationary processes." *The annals of Statistics*, **25**(1), 1–37. Dahlhaus R, Giraitis L (1998). "On the optimal segment length for parameter estimates for locally stationary time series." *Journal of Time Series Analysis*, **19**(6), 629–655.

### See Also

`arima.sim`

### Examples

```
block.smooth.periodogram(malleco)
```

---

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

---

### Description

Plots the p-values Ljung-Box test.

### Usage

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

### Arguments

| | |
|---|---|
| `z` | (type: numeric) data vector |
| `lag` | (type: numeric) the number of periods for the autocorrelation |
| `main` | (type: character) a title for the returned plot |

## 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*. `LSTS_lbtp` computes $q$ and returns the p-values graph with lag *j*.

## Value

A ggplot object.

## References

For more information on theoretical foundations and estimation methods see Brockwell PJ, Davis RA, Calder MV (2002). *Introduction to time series and forecasting*, volume 2. Springer. Ljung GM, Box GE (1978). "On a measure of lack of fit in time series models." *Biometrika*, **65**(2), 297–303.

## See Also

`periodogram`

## Examples

```
Box.Ljung.Test(malleco, lag = 5)
```

---

`hessian` *Hessian Matrix*

---

## Description

Numerical aproximation of the Hessian of a function.

## Usage

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

## Arguments

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

## Details

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

## Value

An $n \times n$ matrix of 2nd derivatives, where $n$ is the length of `x0`.

## See Also

`arima.sim`

## Examples

```
# 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)))
}
sqrt(c(var(malleco) / length(malleco), diag(solve(hessian(
  f = loglik, x = mean(malleco), series = malleco,
  sd = sd(malleco)
)))))
```

---

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

## Value

A list with:

| | |
|---|---|
| `residuals` | standard residuals. |
| `fitted_values` | model fitted values. |
| `delta` | variance prediction error. |

## References

For more information on theoretical foundations and estimation methods see Brockwell PJ, Davis RA, Calder MV (2002). *Introduction to time series and forecasting*, volume 2. Springer. Palma W (2007). *Long-memory time series: theory and methods*, volume 662. John Wiley \& Sons. Palma W, Olea R, Ferreira G (2013). "Estimation and forecasting of locally stationary processes." *Journal of Forecasting*, **32**(1), 86–96.

## Examples

```
fit_kalman <- LS.kalman(malleco, start(malleco))
```

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

## Description

Produces a summary of the results to Whittle estimator to Locally Stationary Time Series (`LS.whittle` function).

## Usage

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

## Arguments

| | |
|---|---|
| `object` | (type: list) the output of `LS.whittle` function |

## Details

Calls the output from `LS.whittle` and computes the standard error and p-values to provide a detailed summary.

## 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. |

## See Also

`LS.whittle`

## Examples

```
fit_whittle <- LS.whittle(
  series = malleco, start = c(1, 1, 1, 1),
  order = c(p = 1, q = 0), ar.order = 1, sd.order = 1, N = 180, n.ahead = 10
)
LS.summary(fit_whittle)
```

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 | (type: numeric) univariate time series. |
| start | (type: numeric) numeric vector, initial values for parameters to run the model. |
| order | (type: numeric) vector corresponding to ARMA model entered. |
| ar.order | (type: numeric) AR polimonial order. |
| ma.order | (type: numeric) MA polimonial order. |
| sd.order | (type: numeric) polinomial order noise scale factor. |
| d.order | (type: numeric) d polinomial order, where d is the ARFIMA parameter. |
| include.d | (type: numeric) logical argument for ARFIMA models. If include.d=FALSE then the model is an ARMA process. |
| N | (type: numeric) 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 | (type: numeric) value corresponding to the lag with which will go taking the blocks or windows. |

| | |
|---|---|
| include.taper | (type: logical) logical argument that by default is TRUE. See periodogram. |
| control | (type: list) A list of control parameters. More details in nlminb . |
| lower | (type: numeric) lower bound, replicated to be as long as start. If unspecified, all parameters are assumed to be lower unconstrained. |
| upper | (type: numeric) upper bound, replicated to be as long as start. If unspecified, all parameters are assumed to be upper unconstrained. |
| m | (type: numeric) truncation order of the MA infinity process, by default $m = 0.25n^{0.8}$. Parameter used in LSTS_kalman. |
| n.ahead | (type: numeric) 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 $\theta$ 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. |
| aic | Akaike'S 'An Information Criterion', for one fitted model LS-ARMA or LS-ARFIMA. The formula is $-2L + 2k/n$, where $L$ represents the log-likelihood, $k$ represents the number of parameters in the fitted model and $n$ is 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. |

## See Also

nlminb, LS.kalman

## Examples

```
# Analysis by blocks of phi and sigma parameters
N <- 200
S <- 100
M <- trunc((length(malleco) - N) / S + 1)
table <- c()
for (j in 1:M) {
  x <- malleco[(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)
  )$par)
}
u <- (N / 2 + S * (1:M - 1)) / length(malleco)
table <- as.data.frame(cbind(u, table))
colnames(table) <- c("u", "phi", "sigma")
# Start parameters
phi <- smooth.spline(table$phi, spar = 1, tol = 0.01)$y
fit.1 <- nls(phi ~ a0 + a1 * u, start = list(a0 = 0.65, a1 = 0.00))
sigma <- smooth.spline(table$sigma, spar = 1)$y
fit.2 <- nls(sigma ~ b0 + b1 * u, start = list(b0 = 0.65, b1 = 0.00))
fit_whittle <- LS.whittle(
  series = malleco, start = c(coef(fit.1), coef(fit.2)), order = c(p = 1, q = 0),
  ar.order = 1, sd.order = 1, N = 180, n.ahead = 10
)
```

`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` | (type: numeric) parameter vector. |
| `series` | (type: numeric) univariate time series. |
| `order` | (type: numeric) vector corresponding to `ARMA` model entered. |
| `ar.order` | (type: numeric) AR polimonial order. |
| `ma.order` | (type: numeric) MA polimonial order. |
| `sd.order` | (type: numeric) polinomial order noise scale factor. |
| `d.order` | (type: numeric) `d` polinomial order, where `d` is the `ARFIMA` parameter. |
| `include.d` | (type: numeric) logical argument for `ARFIMA` models. If `include.d=FALSE` then the model is an ARMA process. |
| `N` | (type: numeric) 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` | (type: numeric) value corresponding to the lag with which will go taking the blocks or windows. |
| `include.taper` | (type: logical) 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 $\lambda$ the Fourier frequencies in the block ($2\,\pi\,k/N$, $k = 1,\ldots,N$).

## References

For more information on theoretical foundations and estimation methods see Brockwell PJ, Davis RA, Calder MV (2002). *Introduction to time series and forecasting*, volume 2. Springer. Palma W, Olea R, others (2010). "An efficient estimator for locally stationary Gaussian long-memory processes." *The Annals of Statistics*, **38**(5), 2958–2997.

## See Also

nlminb, LS.kalman

---

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

---

## Description

This function calculates log-likelihood with known $\theta$, 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 | (type: numeric) parameter vector. |
| series | (type: numeric) univariate time series. |
| order | (type: numeric) vector corresponding to ARMA model entered. |
| ar.order | (type: numeric) AR polimonial order. |
| ma.order | (type: numeric) MA polimonial order. |
| sd.order | (type: numeric) polinomial order noise scale factor. |
| d.order | (type: numeric) d polinomial order, where d is the ARFIMA parameter. |
| include.d | (type: numeric) logical argument for ARFIMA models. If include.d=FALSE then the model is an ARMA process. |
| N | (type: numeric) 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 | (type: numeric) value corresponding to the lag with which will go taking the blocks or windows. |
| include.taper | (type: logical) logical argument that by default is TRUE. See periodogram. |
| theta.par | (type: numeric) vector with the known parameters of the model. |

### Details

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

---

## LS.whittle.loglik.theta

*Locally Stationary Whittle Log-likelihood theta*

---

### Description

Calculate the log-likelihood with $\sigma$ 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 | (type: numeric) parameter vector. |
| series | (type: numeric) univariate time series. |
| order | (type: numeric) vector corresponding to ARMA model entered. |
| ar.order | (type: numeric) AR polimonial order. |
| ma.order | (type: numeric) MA polimonial order. |
| sd.order | (type: numeric) polinomial order noise scale factor. |
| d.order | (type: numeric) d polinomial order, where d is the ARFIMA parameter. |
| include.d | (type: numeric) logical argument for ARFIMA models. If include.d=FALSE then the model is an ARMA process. |
| N | (type: numeric) 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 | (type: numeric) value corresponding to the lag with which will go taking the blocks or windows. |
| include.taper | (type: logical) logical argument that by default is TRUE. See periodogram. |
| sd.par | (type: numeric) value corresponding to known variance. |

## Details

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

---

# malleco *Average Araucaria Araucana Tree Ring Width*

---

## Description

A ts object containing average annual ring width measured in milimiters for different Araucaria Araucana trees in the Malleco Region (Chile). The years of observation in this data cover the period 1242-1975.

## Format

A time series object with 734 elements

## Author(s)

National Oceanic and Atmospheric Administration (NOAA)

`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` | (type: numeric) data vector |
| `plot` | (type: logical) logical argument which allows to plot the periodogram. Defaults to TRUE. |
| `include.taper` | (type: logical) 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 $\lambda$ 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. $\lambda$.

## Value

A list with with the periodogram and the lambda values.

## References

For more information on theoretical foundations and estimation methods see Brockwell PJ, Davis RA, Calder MV (2002). *Introduction to time series and forecasting*, volume 2. Springer. Dahlhaus R, others (1997). "Fitting time series models to nonstationary processes." *The annals of Statistics*, **25**(1), 1–37.

## See Also

`fft`, `Mod`, `smooth.spline`.

## Examples

```
# AR(1) simulated
set.seed(1776)
ts.sim <- arima.sim(n = 1000, model = list(order = c(1, 0, 0), ar = 0.7))
per <- periodogram(ts.sim)
per$plot
```

---

`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` | (type: numeric) data vector. |
| `plot` | (type: logical) logical argument which allows to plot the periodogram. Defaults to TRUE. |
| `spar` | (type: numeric) 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`). $\lambda$ 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 with the smooth periodogram and the lambda values

## See Also

`smooth.spline`, `periodogram`

## Examples

```
# AR(1) simulated
require(ggplot2)
set.seed(1776)
ts.sim <- arima.sim(n = 1000, model = list(order = c(1, 0, 0), ar = 0.7))
per <- periodogram(ts.sim)
aux <- smooth.periodogram(ts.sim, plot = FALSE, spar = .7)
sm_p <- data.frame(x = aux$lambda, y = aux$smooth.periodogram)
sp_d <- data.frame(
  x = aux$lambda,
  y = spectral.density(ar = 0.7, lambda = aux$lambda)
)
g <- per$plot
g +
  geom_line(data = sm_p, aes(x, y), color = "#ff7f0e") +
  geom_line(data = sp_d, aes(x, y), color = "#d31244")
```

---

# spectral.density *Spectral Density*

---

## Description

Returns theoretical spectral density evaluated in ARMA and ARFIMA processes.

## Usage

```
spectral.density(ar = numeric(), ma = numeric(), d = 0, sd = 1, lambda = NULL)
```

## Arguments

| | |
|---|---|
| ar | (type: numeric) AR vector. If the time serie doesn't have AR term then omit it. For more details see the examples. |
| ma | (type: numeric) MA vector. If the time serie doesn't have MA term then omit it. For more details see the examples. |
| d | (type: numeric) Long-memory parameter. If d is zero, then the process is ARMA(p,q). |
| sd | (type: numeric) Noise scale factor, by default is 1. |
| lambda | (type: numeric) $\lambda$ parameter on which the spectral density is calculated/computed. If `lambda=NULL` then it is considered a sequence between 0 and $\pi$. |

## Details

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

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

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

## Value

An unnamed vector of numeric class.

## References

For more information on theoretical foundations and estimation methods see Brockwell PJ, Davis RA, Calder MV (2002). *Introduction to time series and forecasting*, volume 2. Springer. Palma W (2007). *Long-memory time series: theory and methods*, volume 662. John Wiley \& Sons.

## Examples

```
# Spectral Density AR(1)
require(ggplot2)
f <- spectral.density(ar = 0.5, lambda = malleco)
ggplot(data.frame(x = malleco, y = f)) +
  geom_line(aes(x = as.numeric(x), y = as.numeric(y))) +
  labs(x = "Frequency", y = "Spectral Density") +
  theme_minimal()
```

---

# ts.diag *Diagnostic Plots for Time Series fits*

---

## Description

Plot time-series diagnostics.

## Usage

```
ts.diag(x, lag = 10, band = qnorm(0.975)/sqrt(length(x)))
```

## Arguments

| | |
|---|---|
| x | (type: numeric) residuals of the fitted time series model. |
| lag | (type: numeric) maximum lag at which to calculate the acf and Ljung-Box test. By default set to 10. |
| band | (type: numeric) absolute value for bandwidth in the the ACF plot. By default set to 'qnorm(0.975)/sqrt(n)' which approximates to 0.07 for malleco data (n = 734) |

## 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

A ggplot object.

## See Also

Box.Ljung.Test

## Examples

```
ts.diag(malleco)
```

# Index