Package ‘PowerSpectrum’ documentation

of

August 26, 2009

Type Package

Title Spectral Analysis of Time Series

Version 0.3

Date 2009-08-18

Author Dmitry Vyushin, Josh Mayer, Paul Kushner

Maintainer Dmitry Vyushin <dmitry.vyushin@utoronto.ca>

Depends R (>= 2.2.0), fracdiff

Description Periodogram and multitaper estimation of univariate time series power spectrum, multitaper cross spectrum estimation, Detrended Fluctuation Analysis, Geweke-Porter-Hudak Estimator, Gaussian Semiparametric Estimator, convergence test and bias and standard deviation test for the Hurst exponent estimators, spectral goodness-of-fit test, Portmanteau tests, estimation of a time series linear trend with its confidence intervals based on white noise, AR(1), and power-law models for the residuals.

License GPL-2

LazyLoad yes

LazyData yes

R topics documented:

`cse` *Cross Spectrum Estimate Object*

Description

Cross spectrum estimate object is generated by the cross spectrum estimation function `cs.mtm` and can be visualized using `plot` function which actually calls `plot.cse`.

Value

An object of class `cse` has the following properties:

`frequency`	a vector of frequencies.
`cross.spectrum`	a multitaper cross-spectrum estimate.
`coherence`	a multitaper spectral coherence estimate.
`coherence.ci`	a jackknifed spectral coherence standard deviation estimate.
`amplitude`	a multitaper amplitude spectrum estimate.
`phase`	a multitaper phase spectrum estimate.
`phase.ci`	a jackknifed phase spectrum standard deviation estimate.
`ntaper`	a number of tapers used in the spectrum estimate.

`series`	a name of the time series.
`taper`	The data taper used
`weight`	The spectrum weighting used
`method`	the type of spectrum estimation method used, in this case `Multitaper`.
`call`	a matched call.

See Also

`cs.mtm`, `plot.cse`, `ps.mtm`

`cs.mtm` *Multitaper Cross-Spectrum Estimator*

Description

This function estimates the cross-spectrum of two given time series using K tapers [1-5]. The DPSS tapers can be used with the adaptive or simple uniform weighting [1,5]. The "sine" tapers are implemented only with the uniform weighting [4]. `cs.mtm` outputs spectral coherence, amplitude spectrum, and phase spectrum estimates and their standard deviations obtained using a jackknife method [2-3]. The output can be visualized using `plot` function which actually calls `plot.cse`.

Usage

```
cs.mtm(x, y, dt = c("dpss", "sine"), wt = c("adapt", "uniform"),
       K = 3, cl = 0.95, isc.cl = c(0.1,0.5,0.9), verbose = TRUE,
       na.action = na.fail, demean = TRUE, series = NULL, ...)
```

Arguments

`x`	a vector containing a uniformly sampled real valued time series.
`y`	a vector containing a uniformly sampled real valued time series.
`dt`	a data taper to be used. If equals to either "dpss" or "sine" then the appropriate taper will be created by a call to `dpss.taper` or `sine.taper` respectively. If of class `dpss.taper` or `sine.taper` or a matrix of size Nx`K` where N is the input time series length and `K` is the number of tapers then `dt` will be used directly.
`wt`	a weighting to use during spectrum estimation. If `dt` is a "sine" taper or a NxK matrix it will be forced to use uniform weighting. In case of the "dpss" taper the adaptive weighting (see [1,5]) can also be used.
`K`	a number of tapers to be used.
`cl`	a confidence level used for power spectrum confidence intervals estimation.
`isc.cl`	a confidence level for independent time series coherence confidence intervals estimation.
`verbose`	a logical flag. If TRUE (the default), prints information while executing.

`na.action`	function to be called to handle missing values.
`demean`	a logical flag. If TRUE (the default), the mean value of `x` is set to 0.
`series`	a name for the series. Default: c(deparse(substitute(x)), deparse(substitute(y))).
`...`	Additional arguments passed to either `dpss.taper` or `sine.taper`, the most useful of which is `K`, the number of data tapers to use.

Value

An object of class `cse` with the following values set:

`frequency`	a vector of frequencies.
`cross.spectrum`	a multitaper cross-spectrum estimate.
`coherence`	a multitaper spectral coherence estimate.
`coherence.ci`	a jackknifed spectral coherence standard deviation estimate.
`amplitude`	a multitaper amplitude spectrum estimate.
`phase`	a multitaper phase spectrum estimate.
`phase.ci`	a jackknifed phase spectrum standard deviation estimate.
`ntaper`	a number of tapers used in the spectrum estimate.
`series`	a name of the time series.
`taper`	a data taper used
`weight`	a spectrum weighting used
`method`	a type of spectrum estimation method used, in this case `Multitaper`.
`call`	the matched call for `cs.mtm`.

References

[1] D.J. Thomson (1982), Spectrum estimation and harmonic analysis. *Proc. IEEE* **70**, 1055-1096.

[2] D.J. Thomson and A. D. Chave (1991), Jackknifed error estimates for spectra, coherences, and transfer functions, in *Advances in Spectrum Analysis and Array Processing*, S. Haykin, Ed. Englewood Cliffs, NJ: Prentice-Hall, vol. 1, ch. 2, pp. 58–113.

[3] F.L. Vernon et al. (1991), Coherence of seismic body waves from local events as measured by a small-aperture array, *J. Geophys. Res.* **96**, 11981-11996.

[4] K. S. Riedel and A. Sidorenko (1995), Minimum bias multiple taper spectral estimation, *IEEE Transactions on Signal Processing* **43**, 188-195.

[5] D.J. Thomson, L.J. Lanzerotti, F.L. Vernon, M.R. Lessard, and L.T.P. Smith (2007), Solar Modal Structure of the Engineering Environment, *Proc. IEEE* **95**, 1085-1132.

See Also

`plot.cse`, `dpss.taper`, `sine.taper`, `ps.mtm`

Examples

```
library(PowerSpectrum)
Period = seq((1856-1659+1), length(CET_1659_2008))
CET_1856_2008 = CET_1659_2008[Period]
x = cs.mtm(CET_1856_2008, AMO_1856_2008)
plot(x)
```

---

data.update *Climatic Time Series Update*

Description

This procedure downloads recent updates of most of the climatic time series included into the package. It can also save these time series in corresponding rda (R-Data) files in a local folder.

Usage

```
data.update(save = FALSE)
```

Arguments

save	a logical flag. If TRUE, the downloaded climatic time series are saved in corresponding rda (R-Data) files in a local folder. The default is FALSE.

See Also

ps.data

---

dfa.ffe *Detrended Fluctuation Analysis*

Description

Detrended Fluctuation Analysis (DFA) was originally proposed in [1] and is described in details in [2]. It works as follows. In the beginning a cumulative sum time series is generated from the original time series. It might be thought as a random walk which increments are equal to the values of the original time series. Then the cumulative time series is split into segments of size s and is approximated in the least squares sense in each segment by a polynomial of a certain $order$. In most cases $order$ is chosen between 1 and 5. The standard deviation of the best fit residuals is calculated for each segment and then averaged over all segments. Let's call this value $F(s)$. After that the segment size is increased and the above described procedure is repeated. Therefore for each value of s we obtain a corresponding value $F(s)$, which is called fluctuation function. This function estimates $F(s)$.

In case time series autocorrelation function decays as at^{2H-2} when $t \to \infty$ or equivalently its spectral density increases as $b\lambda^{1-2H}$ when $\lambda \to 0$ its fluctuation function $F(s)$ scales as rs^H (see

[3,4]). Thus to extract the Hurst exponent $F(s)$ could be regressed against a straight line in log-log coordinates from the lower scale L to the maximum scale M (as in [1,2]). This regression is done by the `dfa.lse` function.

The output can be visualized using `plot` function which actually calls `plot.ffe`.

Usage

```
dfa.ffe(x, order = 1, verbose = TRUE, na.action = na.fail,
    demean = TRUE, series = NULL, ...)
```

Arguments

`x`	a vector containing a uniformly sampled real valued time series.
`order`	an order of the polynomials used in local detrending. It should be between 1 and 5.
`verbose`	a logical flag. If TRUE (the default), prints information while executing.
`na.action`	a function to be called to handle missing values.
`series`	a name for the time series. Default: deparse(substitute(x)).
`demean`	a logical flag. If TRUE (the default), the mean value of `x` is set to 0.
`...`	

Value

an object of class `ffe` with the following values set:

`fluctuation`	a fluctuation function.
`scale`	a vector of scales.
`order`	the order of the polynomials used in local detrending.
`method`	a fluctuation function estimation method used, in this case "Detrended Fluctuation Analysis".
`series`	a name of the time series. Default: deparse(substitute(x)).
`call`	the matched call to `dfa.ffe`

References

[1] C. Peng, C., S. Buldyrev, A. Goldberger, S. Havlin, M. Simons, and H. Stanley (1993), Finite-size effects on long-range correlations: Implications for analyzing dna sequences, *Phys. Rev. E* **47**, 3730–3733.

[2] J. Kantelhardt, E. Koscielny-Bunde, H. Rego, S. Havlin, and A. Bunde (2001), Detecting long-range correlations with detrended fluctuation analysis, *Physica A* **295**, 441–454.

[3] M. Taqqu, V. Teverovsky, and W. Willinger (1995), Estimators for long-range dependence: an empirical study, *Fractals* **3**, 785–798.

[4] C. Heneghan and G. McDarby (2000), Establishing the relation between detrended fluctuation analysis and power spectral density analysis for stochastic processes. *Phys. Rev. E* **62**, 6103–6110.

See Also

ffe, dfa.lse, plot.ffe

Examples

```
library(PowerSpectrum)
x = dfa.ffe(CET_1659_2008)
plot(x)
```

---

dfa.lse   *Detrended Fluctuation Analysis*

Description

Detrended Fluctuation Analysis (DFA) was originally proposed in [1] and is described in details in [2]. It works as follows. In the beginning a cumulative sum time series is generated from the original time series. It might be thought as a random walk which increments are equal to the values of the original time series. Then the cumulative time series is split into segments of size s and is approximated in the least squares sense in each segment by a polynomial of a certain $order$. In most cases $order$ is chosen between 1 and 5. The standard deviation of the best fit residuals is calculated for each segment and then averaged over all segments. Let's call this value $F(s)$. After that the segment size is increased and the above described procedure is repeated. Therefore for each value of s we obtain a corresponding value $F(s)$, which is called fluctuation function. $F(s)$ is estimated by `dfa.ffe`.

In case time series autocorrelation function decays as at^{2H-2} when $t \to \infty$ or equivalently its spectral density increases as $b\lambda^{1-2H}$ when $\lambda \to 0$ its fluctuation function $F(s)$ scales as rs^H (see [3,4]). Thus to extract the Hurst exponent $F(s)$ is regressed against a straight line in log-log coordinates from the lower scale L to the maximum scale M (as in [1,2]). This regression is done by this function.

The output can be visualized using `plot` function which actually calls `plot.ffe`.

Usage

```
dfa.lse(x, L = (3*x$order+9), M = round(x$scale[length(x$scale)]/4),
        verbose = TRUE, ffe = NULL, ...)
```

Arguments

`x`	an object of class `ffe`
`L`	a lower scale cut off.
`M`	an upper scale cut off.
`verbose`	a logical flag. If TRUE (the default), prints information while executing.
`ffe`	the ffe object used
`...`	

Value

An object of class `tdhee` with the following values set

`H`	an estimate of the Hurst exponent.
`stdH`	a standard deviation of the estimator of `H`.
`r`	a fluctuation function scaling factor from $F(s) \sim rs^H$.
`q`	$q = log(r)$.
`stdq`	a standard deviation of the estimate of `q`.
`L`	a lower scale cut off.
`M`	an upper scale cut off.
`ffe`	the name of the ffe object used. Default: deparse(substitute(x)).
`method`	a Hurst exponent estimation method used, in this case "Least Squares Estimate".
`call`	the matched call to `dfa.lse`

Note

`stdH` and `stdq` are estimated using a crude assumption that the residuals of the linear regression of a DFA curve in log-log coordinates are independent and normally distributed. Thus these values just give an idea about the true uncertainties. Unfortunately the theory that would describe distributions of `stdH` and `stdq` is still missing.

References

[1] C. Peng, C., S. Buldyrev, A. Goldberger, S. Havlin, M. Simons, and H. Stanley (1993), Finite-size effects on long-range correlations: Implications for analyzing dna sequences, *Phys. Rev. E* **47**, 3730–3733.

[2] J. Kantelhardt, E. Koscielny-Bunde, H. Rego, S. Havlin, and A. Bunde (2001), Detecting long-range correlations with detrended fluctuation analysis, *Physica A* **295**, 441–454.

[3] M. Taqqu, V. Teverovsky, and W. Willinger (1995), Estimators for long-range dependence: an empirical study, *Fractals* **3**, 785–798.

[4] C. Heneghan and G. McDarby (2000), Establishing the relation between detrended fluctuation analysis and power spectral density analysis for stochastic processes. *Phys. Rev. E* **62**, 6103–6110.

See Also

`dfa.ffe`, `ffe`, `tdhee`, `plot.ffe`

Examples

```
library(PowerSpectrum)
cet.dfa.ffe <- dfa.ffe(CET_1659_2008)
cet.dfa.lse <- dfa.lse(cet.dfa.ffe)
plot(cet.dfa.ffe,h=cet.dfa.lse)
```

`dpss.taper` *Computing Thomson's Spectral Multitapers by Inverse Iteration*

Description

The following function links the subroutines in "bell-p-w.o" to an R function in order to compute discrete prolate spheroidal sequences (dpss)

Usage

```
dpss.taper(n, K = 3, nmax = 2^(ceiling(log(n, 2))), ...)
```

Arguments

`n`	length of data taper(s)
`K`	number of data tapers
`nmax`	maximum possible taper length, necessary for FORTRAN code
`...`	

Details

Spectral estimation using a set of orthogonal tapers is becoming widely used and appreciated in scientific research. It produces direct spectral estimates with more than 2 df at each Fourier frequency, resulting in spectral estimators with reduced variance. Computation of the orthogonal tapers from the basic defining equation is difficult, however, due to the instability of the calculations – the eigenproblem is very poorly conditioned. In this article the severe numerical instability problems are illustrated and then a technique for stable calculation of the tapers – namely, inverse iteration – is described. Each iteration involves the solution of a matrix equation. Because the matrix has Toeplitz form, the Levinson recursions are used to rapidly solve the matrix equation. FORTRAN code for this method is available through the Statlib archive. An alternative stable method is also briefly reviewed.

Value

an object of class `dpss.taper` with the following properties:

`eigenvectors`	matrix of data tapers (cols = tapers)
`eigenvalues`	eigenvalue associated with each data taper

Author(s)

B. Whitcher, modified by J. Mayer

References

B. Bell, D. B. Percival, and A. T. Walden (1993) Calculating Thomson's spectral multitapers by inverse iteration, *Journal of Computational and Graphical Statistics*, **2**, No. 1, 119-130.

Percival, D. B. and A. T. Walden (1993) *Spectral Estimation for Physical Applications: Multitaper and Conventional Univariate Techniques*, Cambridge University Press.

See Also

`sine.taper`.

---

ffe — *Fluctuation Function Estimate Object*

Description

Fluctuation function estimate object is generated by `dfa.ffe` and is used as an input into `dfa.lse`. The `ffe` object can be visualized using `plot` function which actually calls `plot.ffe`.

Value

An object of class `ffe` has the following properties:

`fluctuation`	a fluctuation function.
`scale`	a vector of scales.
`order`	the order of the polynomials used in local detrending.
`method`	a fluctuation function estimation method used.
`series`	a name of the time series.
`call`	a matched call.

See Also

`dfa.ffe`, `dfa.lse`, `plot.ffe`

gfit.test *Test of the goodness-of-fit tests*

Description

This functions performs a Monte-Carlo kind of test of the two goodness-of-fit tests, Ljung-Box (see `pmt.test`) and spectral density (see `sdf.test`) tests [1-2]. It generates s time series of length n using a power law and an AR(1) models. Then it fits the power law time series by an AR(1) model and the AR(1) time series by a power law model and estimates the probability of rejecting the null hypothesis of a "true" model by the two goodness-of-fit tests. The `gfit.test` replicates the procedure described in [2] using functions implemented in this R package.

Usage

```
gfit.test(H = 0.8, phi = 0.5, sd.fd.res = 1, sd.ar.res = 1,
          lfc = 0, hfc =2, s = 100, n = seq(400,2000,100),
          verbose = TRUE, plot = TRUE)
```

Arguments

`H`	a Hurst exponent value to be tested.
`phi`	a lag one autocorrelation value to be tested.
`sd.fd.res`	the standard deviation of the fractionally differenced process.
`sd.ar.res`	the standard deviation of the AR(1) process.
`lfc`	a number of the lowest Fourier frequences trimmed. Used in `sdf.test` only.
`hfc`	a lower scale cut off. Thus M=trunc(n[i]/hfc). Used in `sdf.test` only.
`s`	the number of samples to average over.
`n`	a vector of time series lengths.
`verbose`	a logical flag. If TRUE (the default), prints information while executing.
`plot`	a logical value for whether or not to plot the results. Default: TRUE.

Value

A list of class `Gtest` with the following elements:

`p`	An array of probabilities of rejecting the null hypothesis that a fitted model (AR(1) or Power Law) is adequate for a realization of a process (Power Law or AR(1)) using the Ljung-Box and Spectral density tests. The output can be visualized using `plot` function which actually calls `plot.gfit.test`.

References

[1] J. Beran (1992), A Goodness-of-Fit Test for Time Series with Long Range Dependence, *J.R. Statis. Soc. B* **54**, 749-760.

[2] D.B. Percival, J.E. Overland, and H.O. Mofjeld (2001), Interpretation of North Pacific Variability as a Short- and Long-Memory Process, *J. Climate* **14**, 4545-4559.

See Also

pmt.test, sdf.test

Examples

```
library(PowerSpectrum)
gfit.test(s=10, n = seq(400,1000,100))
```

---

hurst.conv — *Test of the Hurst exponent estimators convergence*

Description

This function estimates the biases of a given list of the Hurst exponent estimators for a given set of time series lengths for a fixed value of H. Synthetic time series are generated using ARFIMA(0,$H-$0.5,0) model (`fracdiff.sim` function from **fracdiff** package). Results can be nicely plotted using the `plot` function.

Usage

```
hurst.conv(H = 0.8, T = seq(270,910,by=90), s = 100, order = 3,
           lfc = 0, hfc = 18, methods = c("dfa.lse", "pgram.gphe",
           "mtm.gphe", "pgram.gspe", "mtm.gspe"),
           verbose = TRUE, plot = TRUE, ...)
```

Arguments

`H`	a value of the Hurst exponent to be tested, where $0 < H < 2$.
`T`	a vector of time series lengths.
`s`	a number of samples to use.
`order`	the order of the polynomials used in local detrending in DFA
`lfc`	a number of the lowest Fourier frequences trimmed. In case lfc=0 then dfa.M=T[i], otherwise ps.L=lfc and dfa.M=round(T[i]/lfc).
`hfc`	a lower scale cut off. Thus dfa.L=hfc and ps.M=trunc(T[i]/hfc).
`methods`	a character string list specifying the methods for the Hurst exponent estimation. Default: c("dfa", "pgramgphe", "mtmgphe", "pgramgspe", "mtmgspe").
`verbose`	a logical flag. If TRUE (the default), prints information while executing.
`plot`	a logical value for whether or not to plot the results. Default: TRUE.
`...`	Additional arguments passed to any of `dfa.ffe`, `dfa.lse`, `ps.pgram`, `ps.mtm`, `ps.gphe`, `ps.gspe` and `plot`. Note: Neither `m` (`dfa.lse`) nor `M` (`ps.gphe` and `ps.gspe`) should be set since they depend on the length of the time series and are therefore generated accordingly.

Value

A list of class `Hconv` with the following elements:

- `H` — a true value of the Hurst exponent tested.
- `T` — a vector of time series lengths.
- `methods` — a character string list specifying the methods for the Hurst exponent estimation.
- `bH` — a bias of the estimated `H`. It is a matrix of size `length(methods)xlength(T)` which (i, j) element is equal to the bias of the method number i for the time series length number j.

Note

To get a clear distinction between the different methods set s at least equal to 1000.

See Also

`dfa.ffe`, `ps.pgram`, `ps.mtm`, `ps.gphe`, `ps.gspe`, `plot.Hconv`

Examples

```
library(PowerSpectrum)
hurst.conv(s=10)
```

`hurst.test` *Test of the Hurst exponent estimators bias and standard deviation*

Description

This function generates s time series of length n using ARFIMA(0,$H-0.5$,0) model (`fracdiff.sim` function from **fracdiff** package) for a vector of the values of H and calculate the bias and the standard deviation for a given list of the Hurst exponent estimators.

Usage

```
hurst.test(H = seq(0.5,1.1,by=0.1), T = 540, s = 100, order=3, lfc=0,
  hfc=18, methods = c("dfa.lse","pgram.gphe","mtm.gphe",
  "pgram.gspe","mtm.gspe"), verbose = TRUE, plot = TRUE, ...)
```

Arguments

- `H` — a vector of the Hurst exponent values to be tested, where $0 < H[*] < 2$.
- `methods` — a character string list specifying the methods for the Hurst exponent estimation. By default it includes all the supported methods.
- `T` — a length of the time series to use.
- `s` — a number of samples to use.
- `order` — the order of the polynomials used in local detrending in DFA

`lfc`	a number of the lowest Fourier frequences trimmed. Thus ps.L=lfc and dfa.M=round(T[i]/lfc). In case lfc=0 then dfa.M=T[i].
`hfc`	a lower scale cut off. Thus dfa.L=hfc and ps.M=trunc(T[i]/hfc).
`verbose`	a logical flag. If TRUE (the default), prints information while executing.
`plot`	a logical value for whether or not to plot the results. Default: TRUE.
`...`	Additional arguments passed to any of `dfa.ffe`, `dfa.lse`, `ps.pgram`, `ps.mtm`, `ps.gphe`, `ps.gspe` and `plot`.

Value

A list of class `Htest` with the following elements:

`H`	a vector of the true values of the Hurst exponent tested.
`bH`	a bias of the estimated `H`. It is a matrix of size `length(methods)xlength(H)` which (i, j) element is equal to the bias of the method number i for the true value of `H` number j.
`sdH`	a standard deviation of the estimated `H`. It has the same structure as `bH`.

Note

To get a clear distinction between the different methods set s at least equal to 1000.

See Also

`dfa.ffe`, `ps.pgram`, `ps.mtm`, `ps.gphe`, `ps.gspe`, `plot.Htest`

Examples

```
library(PowerSpectrum)
hurst.test(s=10)
```

---

plot.cse *Function for plotting objects of class cse*

Description

This function plots spectral coherence, amplitude spectrum, and phase spectrum estimated by the multitaper method for two time series. It takes as input an object of class `cse`, which can be generated, for instance, by the `cs.mtm` function.

Usage

```
## S3 method for class 'cse':
plot(x, type = "o", main = rep(NULL,3), xlab = rep(NULL,3),
            ylab = rep(NULL,3), plot.ci = TRUE, ...)
```

Arguments

`x`	an object of class `cse`.
`type`	a type of curve used in the plot. See `type` option of the `plot` function.
`main`	a main title of the plot.
`xlab`	a label for the x axis, defaults to a description of x.
`ylab`	a label for the y axis, defaults to a description of y.
`plot.ci`	a logical flag. If TRUE (the default), include confidence intervals in plot
`...`	Additional arguments passed to `plot`.

See Also

`cs.mtm`

Examples

```
library(PowerSpectrum)
Period = seq((1856-1658), length(CET_1659_2008))
CET_1856_2008 = CET_1659_2008[Period]
x <- cs.mtm(CET_1856_2008, AMO_1856_2008)
plot(x)
```

`plot.ffe` *Visualisation of Detrended Fluctuation Analysis*

Description

Function for plotting objects of class `ffe` generated by the PowerSpectrum package.

Usage

```
## S3 method for class 'ffe':
plot(x, h = NULL, plot.ci = TRUE, type = "o", xlim = NULL,
     ylim = NULL, main = NULL, xlab = NULL, ylab = NULL, ...)
```

Arguments

`x`	an object of class `ffe`.
`h`	optional object of class `tdhee`. It could be generated by `dfa.lse`. Adds a fitted power law spectral density to the plot.
`plot.ci`	a logical value for whether or not to plot confidence intervals for a fluctuation function approximation. Default: TRUE.
`type`	a type of curve used in the plot. See `type` option of the `plot` function.
`xlim, ylim`	numeric vectors of length 2, giving the x and y coordinates ranges.
`main`	a main title for the plot.
`xlab`	a label for the x axis, defaults to a description of `x`.
`ylab`	a label for the y axis, defaults to a description of `y`.
`...`	

See Also

dfa.ffe

Examples

```
library(PowerSpectrum)
cet.dfa <- dfa.ffe(CET_1659_2008)
plot(cet.dfa)
```

---

plot.Gtest *Plot of the goodness-of-fit tests results*

Description

Function for plotting objects of class `Gtest` generated by `gfit.test`.

Usage

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

Arguments

x	an object of class `Gtest` generated by `gfit.test`.
...	

See Also

gfit.test

Examples

```
library(PowerSpectrum)
Gtest <- gfit.test(s=10, n = seq(400,1000,100), plot = FALSE)
plot(Gtest)
```

plot.Hconv	*Plot of the Hurst exponent estimators convergence test*

Description

Function for plotting objects of class `H.conv` generated by `hurst.conv`.

Usage

```
## S3 method for class 'Hconv'
plot(x, plot.color = TRUE, ...)
```

Arguments

`x`	an object of class `Hconv` generated by `hurst.conv`.
`plot.color`	a logical value for whether or not to plot with color. Default: TRUE.
`...`	

See Also

`hurst.conv`

Examples

```
library(PowerSpectrum)
conv <- hurst.conv(s=10, plot = FALSE)
plot(conv)
```

plot.Htest	*Plot of the Hurst exponent estimators bias and standard deviation test*

Description

This function plots objects of class `Htest` generated by `hurst.test`.

Usage

```
## S3 method for class 'Htest'
plot(x, plot.panel = 2, plot.color = TRUE, ...)
```

Arguments

`x`	an object of class `Htest` generated by `hurst.test`.
`plot.panel`	an integer value of 1 or 2, determining whether to make a one panel or two panel plot. Default: 2.
`plot.color`	a logical value for whether or not to plot with color. Default: TRUE.
`...`	

See Also

hurst.test

Examples

```
library(PowerSpectrum)
test <- hurst.test(s=10, plot = FALSE)
plot(test)
```

---

plot.pse — *Function for plotting objects of class pse, generated by the PowerSpectrum package.*

Description

This function produces a plot of a power spectrum estimate and its approximations.

Usage

```
## S3 method for class 'pse':
plot(x, ar = NULL, h = NULL, plot.ci = TRUE, type = "o",
         xlim = NULL, ylim = NULL, main = NULL, xlab = NULL,
         ylab = NULL, xaxt="s", ...)
```

Arguments

`x`	an object of class `pse`.
`ar`	optional object of class `sdare`. It could be generated by `ps.ar1`. Adds a fitted AR1 spectral density to the plot.
`h`	optional object of class `sdhee`. It could be generated by `ps.gphe` or `ps.gspe`. Adds a fitted power law spectral density to the plot.
`plot.ci`	a logical flag. If TRUE (the default), include confidence intervals in plot
`type`	a type of curve used in the plot. See `type` option of the `plot` function.
`xlim, ylim`	Numeric vectors of length 2, giving the x and y coordinates ranges.
`main`	a main title of the plot.
`xlab`	a label for the x axis, defaults to a description of `x`.
`ylab`	a label for the y axis, defaults to a description of `y`.
`xaxt`	a character which specifies the x axis type. Specifying "n" suppresses plotting of the axis. The default value is "s".
`...`	

See Also

ps.pgram, ps.mtm, ps.gphe, ps.gspe

Examples

```
library(PowerSpectrum)
pse = ps.pgram(CET_1659_2008)
sdare = ps.ar1(pse)
sdhee = ps.gphe(pse)
plot(pse, sdare, sdhee)
```

`pmt.test` *Portmanteau tests*

Description

The portmanteau test is designed to see if the sample autocorrelations of the residuals for lags t =1,..,lag is consistent with a hypothesis of zero mean white noise, where "lag" is taken to be relatively small in relation to the sample size N. Here we consider two variations on the portmanteau test, namely, the Box-Pierce test statistic and the Ljung-Box-Pierce test statistic [1]. `pmt.test` estimates the residuals for a given sample and a given model, AR1 or power law, and then calls `Box.text` function, which is a standard R function.

Usage

```
pmt.test(x, m, lag = max(10,round(length(x)/20)),
         type = c("Ljung-Box","Box-Pierce"),
         na.action = na.fail, demean = TRUE,
         series = NULL)
```

Arguments

`x`	a vector containing a uniformly sampled real valued time series.
`m`	an object of class `sdhee` (generated by `ps.gphe` or `ps.gspe`) or `sdare` (generated by `ps.ar1`)
`lag`	a maximum autocorrelation function time lag.
`type`	test type, "Ljung-Box" or "Box-Pierce".
`na.action`	a function to be called to handle missing values.
`demean`	a logical flag. If TRUE (the default), the mean value of `x` is set to 0.
`series`	a name for the series. Default: deparse(substitute(x)).

Value

`Bt`	an output of the `Box.text` function.
`model`	a character string specifying the fitted model.

References

[1] D.B. Percival, J.E. Overland, and H.O. Mofjeld (2001), Interpretation of North Pacific Variability as a Short- and Long-Memory Process, *J. Climate* **14**, 4545–4559.

See Also

sdf.test

Examples

```
library(PowerSpectrum)
h <- ps.gspe(ps.mtm(hadcrut3gl_1850_2008))
pmt.test(hadcrut3gl_1850_2008, m=h)
```

ps.ar1 — *Spectral Domain Lag One Autocorrelation (AR1) Estimator*

Description

Spectral domain lag one autocorrelation coefficient estimate object is obtained by fitting the spectral density of AR1 process to an estimate of the power spectrum. It outputs an object of type `sdare`, which serves as an input into a goodness-of-fit test (`sdf.test`), a linear trend test (`trend.test`), and `ps.plot`.

Usage

```
ps.ar1(x, method = c("mle", "lse"), verbose = TRUE, pse = NULL, ...)
```

Arguments

`x`	an object of class `pse`, output from either `ps.pgram` or `ps.mtm`.
`method`	the method used to estimate the lag one autocorrelation coefficient
`verbose`	a logical flag. If TRUE (the default), prints information while executing.
`pse`	the name of the pse object. Default: deparse(substitute(x)).
`...`	

Value

An object of class `sdare` with the following values set:

`phi`	an estimate of the lag one autocorrelation coefficient.
`sdphi`	a standard deviation of the estimator of `phi`.
`pse`	the name of the pse object used.
`method`	a lag one autocorrelation estimation method used.
`call`	the matched call to `ps.ar1`

See Also

sdare, sdf.test, trend.test

Examples

```
library(PowerSpectrum)
ps.ar1(ps.pgram(AMO_1856_2008))
```

`ps.data` *Climatic Time Series*

Description

The list of climatic time series included into the package is shown below. Most of these time series can be updated using `data.update`.

`AMO_1.1856_7.2009` - monthly data of Atlantic Multidecadal Oscillation index
http://www.cdc.noaa.gov/Timeseries/AMO/.

`AMO_1856_2008` - annual data of Atlantic Multidecadal Oscillation index.

`CET_1.1659_7.2009` - monthly data of Central England Temperature
http://hadobs.metoffice.com/hadcet/.

`CET_1659_2008` - annual data of Central England Temperature.

`crutem3gl_1.1850_6.2009` - Land Surface Temperature Anomalies (Global, monthly means)
http://www.cru.uea.ac.uk/cru/data/temperature/

`crutem3gl_1850_2008` - Land Surface Temperature Anomalies (Global, annual means)

`crutem3nh_1.1850_6.2009` - Land Surface Temperature Anomalies (Northern Hemisphere, monthly means)

`crutem3nh_1850_2008` - Land Surface Temperature Anomalies (Northern Hemisphere, annual means)

`crutem3sh_1.1850_6.2009` - Land Surface Temperature Anomalies (Southern Hemisphere, monthly means)

`crutem3sh_1850_2008` - Land Surface Temperature Anomalies (Southern Hemisphere, annual means)

`Donard_752_1992` - Donard Lake (Baffin Island) summer temperature reconstruction based on lake varve thickness
ftp://ftp.ncdc.noaa.gov/pub/data/paleo/paleolimnology/northamerica/canada/baffin/donard_2001.txt

`giss_ghcn_gl_1.1880_12.2008` - GISS Global Temperature Anomalies (base period: 1951-1980, monthly means). Sources: GHCN (meteorological stations only)
http://data.giss.nasa.gov/gistemp/

giss_ghcn_gl_1880_2008 - GISS Global Temperature Anomalies (base period: 1951-1980, annual means). Sources: GHCN (meteorological stations only)

giss_ghcn_nh_1.1880_12.2008 - GISS Northern Hemisphere Temperature Anomalies (base period: 1951-1980, monthly means). Sources: GHCN (meteorological stations only)

giss_ghcn_nh_1880_2008 - GISS Northern Hemisphere Temperature Anomalies (base period: 1951-1980, annual means). Sources: GHCN (meteorological stations only)

giss_ghcn_sh_1.1880_12.2008 - GISS Southern Hemisphere Temperature Anomalies (base period: 1951-1980, monthly means). Sources: GHCN (meteorological stations only)

giss_ghcn_sh_1880_2008 - GISS Southern Hemisphere Temperature Anomalies (base period: 1951-1980, monthly means). Sources: GHCN (meteorological stations only)

giss_ghcn_sst_gl_1.1880_12.2008 - GISS Global Temperature Anomalies (base period: 1951-1980, monthly means). Sources: GHCN + SST.

giss_ghcn_sst_gl_1880_2008 - GISS Global Temperature Anomalies (base period: 1951-1980, annual means). Sources: GHCN + SST.

giss_ghcn_sst_nh_1.1880_12.2008 - GISS Northern Hemisphere Temperature Anomalies (base period: 1951-1980, monthly means). Sources: GHCN + SST.

giss_ghcn_sst_nh_1880_2008 - GISS Northern Hemisphere Temperature Anomalies (base period: 1951-1980, annual means). Sources: GHCN + SST.

giss_ghcn_sst_sh_1.1880_12.2008 - GISS Southern Hemisphere Temperature Anomalies (base period: 1951-1980, monthly means). Sources: GHCN + SST.

giss_ghcn_sst_sh_1880_2008 - GISS Southern Hemisphere Temperature Anomalies (base period: 1951-1980, monthly means). Sources: GHCN + SST.

hadcrut3gl_1.1850_6.2009 - Combined Land and Marine Surface Temperature Anomalies (Global, monthly means)
http://www.cru.uea.ac.uk/cru/data/temperature/

hadcrut3gl_1850_2008 - Combined Land and Marine Surface Temperature Anomalies (Global, annual means)

hadcrut3nh_1.1850_6.2009 - Combined Land and Marine Surface Temperature Anomalies (Northern Hemisphere, monthly means)

hadcrut3nh_1850_2008 - Combined Land and Marine Surface Temperature Anomalies (Northern Hemisphere, annual means)

hadcrut3sh_1.1850_6.2009 - Combined Land and Marine Surface Temperature Anomalies (Southern Hemisphere, monthly means)

`hadcrut3sh_1850_2008` - Combined Land and Marine Surface Temperature Anomalies (Southern Hemisphere, annual means)

`NAO_DJFM_Hurrell_1864_2008` - Jim Hurrell's winter (December through March) index of the NAO based on the difference of normalized sea level pressure (SLP) between Lisbon, Portugal and Stykkisholmur/Reykjavik, Iceland
http://www.cgd.ucar.edu/cas/jhurrell/Data/naodjfmindex.asc.

`PDO_1.1900_6.2009` - monthly data of Pacific Decadal Oscillation index
http://www.jisao.washington.edu/pdo/.

`PDO_1900_2008` - annual data of Pacific Decadal Oscillation index

`Rarotonga_1726_1996` - annual data of Rarotonga coral Sr/Ca SST reconstruction
ftp://ftp.ncdc.noaa.gov/pub/data/paleo/coral/east_pacific/rarotonga_sr-ca.txt.

See Also

`data.update`

`pse` *Power Spectrum Estimate Object*

Description

Power spectrum estimate object is generated by power spectrum estimation functions (`ps.pgram`, `ps.mtm`) and serves as an input into a goodness-of-fit test (`sdf.test`) and functions estimating power spectrum approximations (`ps.ar1`, `ps.gphe`, `ps.gspe`).

Value

An object of class `pse` has the following properties:

`frequency`	a vector of frequencies.
`spectrum`	a power spectrum estimate.
`spectrum.ci`	an asymptotic in case of the periodogram or a jackknifed in case of the multitaper confidence interval for the power spectrum estimate.
`cl`	a confidence level used for power spectrum confidence interval estimation.
`ntaper`	a number of tapers used in the spectrum estimate.
`taper`	The data taper used
`weight`	The spectrum weighting used
`series`	a name of the time series.
`method`	a spectrum estimation method used.
`call`	a matched call for `ps.mtm`.

See Also

ps.pgram, ps.mtm, sdf.test, ps.ar1, ps.gphe, ps.gspe

---

ps.gphe *Geweke-Porter-Hudak Estimator*

Description

Geweke-Porter-Hudak Estimator (GPHE) is a linear fit of the time series power spectrum in log-log coordinates within a given frequency bandwidth. GPHE estimates the Hurst exponent together with its confidence intervals and a scaling factor b by fitting function $f(\lambda) = b|\lambda|^{1-2H}$ to a low-frequency part of the time series power spectrum by the least squares method. GPHE was originally proposed in [1] and rigorously justified in [2] and [3] for the case of the periodogram and in [4] for the multitaper.

Usage

```
ps.gphe(x, L = 0, M = length(f), calcSD = FALSE,
        verbose = TRUE, pse = NULL, ...)
```

Arguments

x	an object of class pse, output from either ps.pgram or ps.mtm.
L	a number of the lowest Fourier frequences trimmed.
M	a number of the highest Fourier frequency used.
calcSD	a logical flag. If TRUE, calculates the standard deviations (see equation (11) in [4]) for the estimates of H and $c = log(b)$. It is a time consuming option. Default: FALSE.
verbose	a logical flag. If TRUE (the default), prints information while executing.
pse	the name of the pse object. Default: deparse(substitute(x)).
...	

Value

An object of class sdhee with the following values set:

H	an estimate of the Hurst exponent.
sdH	a standard deviation of the estimator of H (see equation (11) in [3]). GPHE only.
asdH	an asymptotic value of the standard deviation of the estimator of H based on the periodogram (see equation (7) on page 24 in [2] for GPHE and equation (4.1) on page 1640 in [1] for GSPE).
b	an estimate of the scaling factor b from $f(\lambda) = b\lambda^{1-2H}$.
c	$c = log(b)$.
sdc	a standard deviation of the estimator of $c = log(b)$ (see equation (11) in [3]).

`L`	a number of the lowest Fourier frequences trimmed.
`M`	a number of the highest Fourier frequency used.
`psa`	a power spectrum approximation of the form $b\lambda^{1-2H}$.
`series`	a name of the time series.
`method`	a Hurst exponent estimation method used.
`call`	a matched call.

References

[1] J. Geweke and S. Porter-Hudak (1983), The estimation and application of long-memory time series models, *J. Time Series Anal.* **4**, 221–238.

[2] P.M. Robinson (1995), Log-periodogram regression of time series with long range dependence, *Ann. of Statist.* **23**, 1048–1072.

[3] C. Hurvich, R. Deo, and J. Brodsky (1998), The mean squared error of geweke and porter-hudak's estimator of the memory parameter of a long-memory time series, *J. Time Series Anal.* **19**, 19–46, 10.1111/1467-9892.00075.

[4] E.J. McCoy, A.T. Walden, and D.B. Percival (1998), Multitaper Spectral Estimation of Power Law Processes, *IEEE Transactions on Signal Processing* **46**, 655–668.

See Also

`pse`, `sdhee`, `ps.pgram`, `ps.mtm`, `ps.gspe`

Examples

```
library(PowerSpectrum)
ps.gphe(ps.mtm(Donard_752_1992))
```

`ps.gspe` *Gaussian Semiparametric Estimator*

Description

Gaussian Semiparametric Estimator (GSPE) is a maximum likelihood fit of the time series power spectrum within a given frequency bandwidth. GSPE estimates the Hurst exponent and a scaling factor b by fitting the function $f(\lambda) = b|\lambda|^{1-2H}$ to a low-frequency part of the time series power spectrum by the maximum likelihood method. It was originally proposed in [1] and rigorously justified in [2].

Usage

```
ps.gspe(x, L = 0, M = length(f), interval = c(0,1.5),
        verbose = TRUE, pse = NULL, ...)
```

Arguments

`x`	an object of class `pse`, output from either `ps.pgram` or `ps.mtm`.
`L`	a number of the lowest Fourier frequences trimmed.
`M`	a number of the highest Fourier frequency used.
`interval`	an interval over which to estimate H.
`verbose`	a logical flag. If TRUE (the default), prints information while executing.
`pse`	the name of the pse object. Default: deparse(substitute(x)).
`...`	

Value

An object of class `sdhee` with the following values set:

`H`	an estimate of the Hurst exponent.
`asdH`	an asymptotic value of the standard deviation of the estimator of `H` based on the periodogram (see equation (7) on page 24 in [2] for GPHE and equation (4.1) on page 1640 in [1] for GSPE).
`b`	an estimate of the scaling factor b from $f(\lambda) = b\lambda^{1-2H}$.
`c`	$c = log(b)$.
`L`	a number of the lowest Fourier frequences trimmed.
`M`	a number of the highest Fourier frequency used.
`psa`	a power spectrum approximation of the form $b\lambda^{1-2H}$.
`series`	a name of the time series.
`method`	a Hurst exponent estimation method used.
`call`	a matched call.

References

[1] R. Fox and M. Taqqu (1988), Large sample properties of parameter estimates for strongly dependent stationary gaussian time series, *Ann. of Statist.* **17**, 1749–1766.

[2] P.M. Robinson (1995), Gaussian estimation of long range dependence, *Ann. of Statist.* **23**, 1630–1661.

See Also

`pse`, `sdhee`, `ps.pgram`, `ps.mtm`, `ps.gphe`

Examples

```
library(PowerSpectrum)
ps.gspe(ps.mtm(AMO_1856_2008))
```

ps.mtm *Multitaper Spectrum Estimator*

Description

Multitaper is an average of several direct spectrum estimators, which use a set of orthogonal tapers (for details see [1-6]). The DPSS tapers can be used with the adaptive or simple uniform weighting [1,3,6]. The "sine" tapers are implemented only with the uniform weighting [4]. Confidence intervals are estimated using a jackknife method [2]. The output can be visualized using `plot` function which actually calls `plot.pse`.

Usage

```
ps.mtm(x, dt = c("dpss", "sine"), wt = c("adapt", "uniform"),
       K = 3, cl = 0.95, verbose = TRUE, na.action = na.fail,
       demean = TRUE, series = NULL, ...)
```

Arguments

`x`	a vector containing a uniformly sampled real valued time series.
`dt`	a data taper to be used. If equals to either "dpss" or "sine" then the appropriate taper will be created by a call to `dpss.taper` or `sine.taper` respectively. If of class `dpss.taper` or `sine.taper` or a matrix of size NxK where N is the input time series length and K is the number of tapers then `dt` will be used directly.
`wt`	a weighting to use during spectrum estimation. If `dt` is a "sine" taper or a NxK matrix it will be forced to use uniform weighting. In case of the "dpss" taper the adaptive weighting (see [1,3,6]) can also be used.
`K`	a number of tapers to be used.
`cl`	a confidence level used for power spectrum confidence intervals estimation.
`verbose`	a logical flag. If TRUE (the default), prints information while executing.
`na.action`	function to be called to handle missing values.
`demean`	a logical flag. If TRUE (the default), the mean value of `x` is set to 0.
`series`	a name for the series. Default: deparse(substitute(x)).
`...`	Additional arguments passed to either `dpss.taper` or `sine.taper`, the most useful of which is `K`, the number of data tapers to use.

Value

An object of class `pse` with the following values set:

`frequency`	a vector of frequencies.
`spectrum`	a power spectrum estimate.
`spectrum.ci`	a jackknifed confidence interval for the power spectrum estimate.

`cl`	a confidence level used for power spectrum confidence interval estimation.
`ntaper`	a number of tapers used in the spectrum estimate.
`taper`	The data taper used
`weight`	The spectrum weighting used
`series`	a name of the time series.
`method`	a spectrum estimation method used.
`call`	a matched call for `ps.mtm`.

References

[1] D.J. Thomson (1982), Spectrum estimation and harmonic analysis. *Proc. IEEE*, **70**, 1055-1096.

[2] D.J. Thomson and A. D. Chave (1991), Jackknifed error estimates for spectra, coherences, and transfer functions, in *Advances in Spectrum Analysis and Array Processing*, S. Haykin, Ed. Englewood Cliffs, NJ: Prentice-Hall, vol. 1, ch. 2, pp. 58-113.

[3] D. Percival and A. Walden (1993), *Spectral Analysis for Physical Applications*, Cambridge University Press, 611 pp.

[4] K.S. Riedel and A. Sidorenko (1995), Minimum bias multiple taper spectral estimation, *IEEE Transactions on Signal Processing*, **43**, 188-195.

[5] E.J. McCoy, A.T. Walden, and D.B. Percival (1998), Multitaper Spectral Estimation of Power Law Processes, *IEEE Transactions on Signal Processing, 46*, 655-668.

[6] D.J. Thomson, L.J. Lanzerotti, F.L. Vernon, M.R. Lessard, and L.T.P. Smith (2007), Solar Modal Structure of the Engineering Environment, *Proc. IEEE*, **95**, 1085-1132.

See Also

`pse`, `plot.pse`, `cs.mtm`, `ps.pgram`

Examples

```
library(PowerSpectrum)
x = ps.mtm(Rarotonga_1726_1996)
plot(x)
```

`ps.pgram` *Periodogram Spectrum Estimator*

Description

Periodogram is the simplest power spectrum estimator (for details see [1]). It estimates the power spectrum through the square of absolute value of discrete Fourier transform of the time series divided by the time series length.

Usage

```
ps.pgram(x, cl = 0.95, verbose = TRUE, na.action = na.fail,
              demean = TRUE, series = NULL, ...)
```

Arguments

`x`	a vector containing a uniformly sampled real valued time series.
`cl`	a confidence level used for power spectrum confidence intervals estimation.
`verbose`	a logical flag. If TRUE (the default), prints information while executing.
`na.action`	a function to be called to handle missing values.
`demean`	a logical flag. If TRUE (the default), the mean value of "x" is set to 0.
`series`	a name for the time series. Default: deparse(substitute(x)).
`...`	

Value

An object of class `pse` with the following values set:

`frequency`	a vector of frequencies.
`spectrum`	a power spectrum estimate.
`spectrum.ci`	an asymptotic confidence interval for the power spectrum estimate.
`cl`	a confidence level used for power spectrum confidence interval estimation.
`ntaper`	a number of tapers used in the spectrum estimate.
`series`	a name of the time series.
`method`	a spectrum estimation method used.
`call`	a matched call for `ps.pgram`.

References

[1] D. Percival and A. Walden (1993), *Spectral Analysis for Physical Applications*, Cambridge University Press, 611 pp.

See Also

`pse`, `ps.mtm`, `ps.gphe`, `ps.gspe`

Examples

```
library(PowerSpectrum)
ps.pgram(AMO_1856_2008)
```

sdare *Spectral Domain Lag One Autocorrelation (AR1) Estimate Object*

Description

Spectral domain lag one autocorrelation estimate object is generated by `ps.ar1` by fitting the spectral density of AR1 process to an estimate of the power spectrum. It serves as an input into a goodness-of-fit test (`sdf.test`) and a linear trend test (`trend.test`).

Value

An object of class `sdare` has the following properties:

`phi`	an estimate of the lag one autocorrelation coefficient.
`sdphi`	a standard deviation of the estimator of phi.
`pse`	the name of the pse object used.
`method`	a lag one autocorrelation estimation method used.
`call`	a matched call.

See Also

`ps.ar1`, `sdf.test`, `trend.test`

---

sdf.test *Spectral Goodness-of-Fit Test*

Description

This test compares a given estimate of the spectrum to the spectral density corresponding to a fitted model, AR1 or a power law, in the frequency range specified by the indices L and M. The null hypothesis is that the AR1 or the power law is a correct model for the given spectrum [1-2]. `sdf.test` outputs the spectral density of the fitted model, a test statistic and a p-value, which is the smallest significance level for which we would end up rejecting the null hypothesis.

Usage

```
sdf.test(x, m, L = 0, M = length(x$frequency), verbose = TRUE)
```

Arguments

`x`	an object of class `pse`. It could be generated by `ps.pgram`.
`m`	an object of class `sdhee` (generated by `ps.gphe` or `ps.gspe`) or `sdare` (generated by `ps.ar1`)
`L`	a number of the lowest Fourier frequences trimmed.
`M`	a number of the highest Fourier frequency used.
`verbose`	a logical flag. If TRUE (the default), prints information while executing.

Value

`T`	the test statistic.
`p`	the test p-value.
`model`	a character string specifying the fitted model.

References

[1] J. Beran (1992), A Goodness-of-Fit Test for Time Series with Long Range Dependence, *J. R. Statis. Soc. B* **54**, 749–760.

[2] D.B. Percival, J.E. Overland, and H.O. Mofjeld (2001), Interpretation of North Pacific Variability as a Short- and Long-Memory Process, *J. Climate* **14**, 4545–4559.

See Also

`pse`, `sdare`, `sdhee`, `pmt.test`, `ps.gphe`, `ps.gspe`

Examples

```
library(PowerSpectrum)
pse = ps.pgram(Rarotonga_1726_1996)
sdare = ps.ar1(pse)
sdhee = ps.gspe(pse)
plot(pse, sdare, sdhee)
sdf.test(pse, m = sdare)
sdf.test(pse, m = sdhee)
```

`sdhee` — *Spectral Domain Hurst Exponent Estimate Object*

Description

Hurst exponent estimate object is generated by Hurst exponent estimation functions (`ps.gphe`, `ps.gspe`) and serves as an input into a goodness-of-fit test (`sdf.test`), a linear trend test (`trend.test`), and `ps.plot`.

Value

An object of class `sdhee` has the following properties:

`H`	an estimate of the Hurst exponent.
`sdH`	a standard deviation of the estimator of H (see equation (11) in [3]). GPHE only.
`asdH`	an asymptotic value of the standard deviation of the estimator of H based on the periodogram (see equation (7) on page 24 in [2] for GPHE and equation (4.1) on page 1640 in [1] for GSPE).
`b`	an estimate of the scaling factor b from $f(\lambda) = b\vert\lambda\vert^{1-2H}$.

`c`	$c = log(b)$.
`sdc`	a standard deviation of the estimator of $c = log(b)$ (see equation (11) in [3]). GPHE only.
`L`	a number of the lowest Fourier frequences trimmed.
`M`	a number of the highest Fourier frequency used.
`series`	a name of the time series.
`method`	a Hurst exponent estimation method used.
`call`	a matched call.

References

[1] P.M. Robinson (1995), Gaussian estimation of long range dependence, *Ann. of Statist.* **23**, 1630–1661.

[2] C. Hurvich, R. Deo, and J. Brodsky (1998), The mean squared error of Geweke and Porter-Hudak's estimator of the memory parameter of a long-memory time series, *J. Time Series Anal.* **19**, 19–46, 10.1111/1467-9892.00075.

[3] E.J. McCoy, A.T. Walden, and D.B. Percival (1998), Multitaper Spectral Estimation of Power Law Processes, *IEEE Transactions on Signal Processing* **46**, 655–668.

See Also

`ps.gphe`, `ps.gspe`, `sdf.test`, `trend.test`

`sine.taper` *Computing Sinusoidal Data Tapers*

Description

Computes sinusoidal data tapers directly from equations.

Usage

```
sine.taper(n, K = 3, ...)
```

Arguments

`n`	length of data taper(s)
`K`	number of data tapers
`...`	

Details

See reference.

Value

an object of class `sine.taper` that is a vector or matrix of data tapers.

Author(s)

B. Whitcher, modified by J. Mayer

References

Riedel, K. S. and A. Sidorenko (1995) Minimum bias multiple taper spectral estimation, *IEEE Transactions on Signal Processing*, **43**, 188-195.

See Also

`dpss.taper`.

`tdhee` *Time Domain Hurst Exponent Estimate Object*

Description

Time domain Hurst exponent estimate object is generated by a time domain Hurst exponent estimation function (`dfa.lse`).

Value

An object of class `tdhee` has the following properties:

`H`	an estimate of the Hurst exponent.
`sdH`	a standard deviation of the estimator of H.
`r`	a fluctuation function scaling factor from $F(s) \sim rs^H$.
`q`	$q = log(r)$.
`sdq`	a standard deviation of the estimate of q.
`L`	a lower scale cut off.
`M`	an upper scale cut off.
`ffe`	the name of the ffe object used.
`method`	a Hurst exponent estimation method used.
`call`	a matched call.

See Also

`dfa.ffe`, `dfa.lse`, `sdhee`

trend.test *Univariate time series linear trend estimation and detection*

Description

This function estimates a linear trend for a univariate time series using linear regression and then estimates its confidence intervals relatively to three competing hypothesis regarding residuals' autocorrelation structure: white noise, AR(1), power law. The function also estimates the number of data points (desired time series length) required to detect the observed trend for a given significance level and a test power under each hypothesis.

Usage

```
trend.test(x, ar = NULL,  h = NULL, a = 0.05, p = 0.5,
           verbose = TRUE, na.action = na.fail,
           demean = TRUE, series = NULL)
```

Arguments

`x`	a vector containing a uniformly sampled real valued time series.
`ar`	optional object of class `sdare`. It could be generated by `ps.ar1`. Tests the trend of AR(1) model.
`h`	optional object of class `sdhee`. It could be generated by `ps.gphe` or `ps.gspe`. Tests the trend of Power Law model.
`a`	significance level
`p`	power of the test specified for calculation of a number of data points required to detect the observed trend
`verbose`	a logical flag. If TRUE (the default), prints information while executing.
`na.action`	function to be called to handle missing values.
`demean`	a logical flag. If TRUE (the default), the mean value of `x` is set to 0.
`series`	a name for the series. Default: deparse(substitute(x)).

Value

A list with the following elements:

`intercept`	an intercept estimate.
`trend`	a slope estimate.
`sd`	a standard deviation of the linear trend residuals.
`trend.est`	a matrix of size 3x2 which first column contains estimated confidence intervals for the trend and the second column contains the number of data points required to detect the estimated trend for the given power. The rows correspond to different assumptions about trend residuals autocorrelation structure. Thus the first row corresponds to the case of white noise residuals, the second to AR1, and the last one to power law.
`length`	the length of the time seires

Note

All periodical signals have to be removed from the time series prior to `trend.test` application!

References

[1] R.L. Smith (1993), Long-range dependence and global warming, In *Statistics for the Environment* (V. Barnett and F. Turkman, eds.), John Wiley, Chichester, 141–161.

[2] D. Vyushin, V. Fioletov, and T. Shepherd, (2007), Impact of long-range correlations on trend detection in total ozone, *J. Geophys. Res.* **112**, 10.1029/2006JD008168, http://www.atmosp.physics.utoronto.ca/people/vyushin/Papers/Vyushin_Fioletov_Shepherd_Trend_Detection_in_Total_Ozone.pdf.

See Also

`sdare`, `sdhee`, `ps.pgram`, `ps.mtm`, `ps.gphe`, `ps.gspe`

Examples

```
library(PowerSpectrum)
NAO = NAO_DJFM_Hurrell_1864_2008[seq((1946-1864+1),(1995-1864+1))]
plot(seq(1946,1995), NAO, type="o", xlab="")
NAOres = lm(NAO ~ seq(1,length(NAO)))$residuals
pse = ps.pgram(NAOres)
sdare = ps.ar1(pse)
sdhee = ps.gspe(pse)
tr = trend.test(NAO, ar=sdare, h=sdhee)
```

Index