Package ‘PaleoSpec’

September 29, 2016

Title Spectral tools for the ECUS group

Version 0.0.0.9000

Description Spectral tools for the ECUS group

Depends R (>= 3.3.1)

License What license is it under?

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

Imports multitaper

R topics documented:

`AddConfInterval` *Add confidence intervals to a spectrum*

Description

Add confidence intervals to a spectrum

Usage

```
AddConfInterval(spec, MINVALUE = 1e-10, pval = 0.05)
```

Arguments

`spec`	spectrum list(spec,freq,dof)
`MINVALUE`	Minimum value to which the confidence interval is limited
`pval`	Interval from (pval/2 to 1-pval/2) is constructed

Value

spectrum as the input but including lim.1 and lim.2 as new list elements

Author(s)

Thomas Laepple

`AnPowerlaw` *A PSD(freq) for a powerlaw with variance 1*

Usage

```
AnPowerlaw(beta, freq, return.scaling = FALSE)
```

Arguments

`beta`	slope of the powerlaw
`freq`	frequency vector

Value

vector containing the PSD

Author(s)

Thomas Laepple

ApplyFilter *Apply a filter to a timeseries*

Description

Apply a filter to a timeseries

Usage

```
ApplyFilter(data, filter, method = 1)
```

Arguments

`data`	Input timeseries (ts object)
`filter`	vector of filter weights
`method`	constraint method choice 1-3

Details

Using endpoint constrains as describen in Mann et al., GRL 2003 minimum norm constraint (method=1) minimum slope constraint (method=2) minimum roughness constraint (method=3)

Value

filtered timeseries (ts object)

Author(s)

Thomas Laepple

Bandpass *calculate weights for a bandpass filter*

Description

based on Bloomfield 1976

Usage

```
Bandpass(omega.upper, omega.lower, n, sample = 1, convergence = T)
```

Arguments

`omega.upper`	upper cutoff frequency
`omega.lower`	lower cutoff frequency
`n`	length of the filter, has to be odd
`sample`	sampling rate of the timeseries on which the filter will be applied
`convergence`	TRUE: smoothed least square lowpass; FALSE = unsmoothed
`omega.c`	cutoff frequency

Value

vector of filter weights

Author(s)

Thomas Laepple

`ColTransparent` *Modify a color to get brighter and tranparent for the confidence intervals*

Description

Modify a color to get brighter and tranparent for the confidence intervals

Usage

```
ColTransparent(color, alpha = 0.8, beta = 150)
```

Arguments

`color`	color value, e.g. "red"
`alpha`	(0..1) transparency value
`beta`	(0..255) to make it brighter, this value gets added on the RGB values

Value

modified color

Author(s)

Thomas Laepple

`ConfRatio` *Confidence Interval of ratios*

Description

Confidence Interval of ratios based on a ChiSquare Distribution

Usage

```
ConfRatio(varratio, df.1, df.2, pval = 0.1)
```

Arguments

`varratio`	
`df.1`	degree of freedom of denominator
`df.2`	degree of freedom of numerator
`pval`	

Value

lower and upper confidence intervals

Author(s)

Thomas Laepple

`ConfVar` *Provide ChiSquared confidence intervals for ratios*

Description

Provide ChiSquared confidence intervals for ratios

Usage

```
ConfVar(varlist, pval = 0.05)
```

Arguments

`varlist`	list(var,dof)
`pval`	requested p-value

Value

Output: confidence intervals

Author(s)

Thomas Laepple

`FirstElement` *first element of a vector*

Usage

```
FirstElement(x)
```

Arguments

`x`

Value

first element of X

Author(s)

Thomas Laepple

`fweights` *weights*

Usage

```
fweights(ftarget, f, df.log)
```

Arguments

`ftarget`

`f`

`df.log`

Value

weight vector

Author(s)

Thomas Laepple

`fweights.lin` *fweights.lin*

Usage

```
fweights.lin(ftarget, f, df.log)
```

Arguments

`ftarget`

`f`

`df.log`

Value

weight vector

Author(s)

Thomas Laepple

`GetTransferFunction` *Derives and plots the transfer function*

Description

Derives and plots the transfer function (given a filter)

Usage

```
GetTransferFunction(g.u, resolution = 100, bPlot = TRUE, add = FALSE, ...)
```

Arguments

`g.u`
`resolution`
`bPlot`
`add`
`...`

Details

Get the transfer function of a symetric filter, page 122 in Bloomfield 1976,

Value

list(omega,y) containing the transfer function

Author(s)

Thomas Laepple

`GetVarFromSpectra` *Variance estimate by integrating a part of the spectrum*

Description

Variance estimate by integrating a part of the spectrum

Usage

```
GetVarFromSpectra(spec, f, dfreq = (f[2] - f[1])/100, df.log = 0, bw = 3)
```

Arguments

`spec`	spectrum (list of spec,freq,dof) to be analysed
`f`	f[1],f[2]: frequency interval to be analysed
`dfreq`	frequency discretisation used in the temporary interpolation
`df.log`	if > 0, smooth the spectra prior to integrating
`bw`	the bandwidth assumed for the confinterval calculation (from the multitaper spectral estimate)

Value

list(var,dof) variance and corresponding dof

Author(s)

Thomas Laepple

Examples

```
x<-ts(rnorm(100))
spec<-SpecMTM(x)
GetVarFromSpectra(spec,c(1/100,0.5))
GetVarFromSpectra(spec,c(0.25,0.5))
```

LastElement *last element of a vector*

Usage

```
LastElement(x)
```

Arguments

x

Value

last element of X

Author(s)

Thomas Laepple

LLines *add Logplot + transparent confidence interval for the spectral plotting*

Description

add Logplot + transparent confidence interval for the spectral plotting

Usage

```
LLines(x, conf = TRUE, col = "black", alpha = 0.3, removeFirst = 0,
  removeLast = 0, ...)
```

Arguments

`x`	spectra object
`conf`	TRUE: Plot confidence interval
`col`	color
`alpha`	transparency
`removeFirst`	omit removeFirst values on the low frequency side
`removeLast`	omit removeFirst values on the high frequency side
`...`	other parameters to be passed to the line function

Value

none

Author(s)

Thomas Laepple

Examples

```
x<-ts(arima.sim(list(ar = 0.9),1000))
spec<-SpecMTM(x)
LPlot(spec,col="grey")
LLines(LogSmooth(spec),lwd=2)
```

LogSmooth *Smoothes the spectrum using a log smoother*

Description

Smoothes the spectrum using a log smoother.

Usage

```
LogSmooth(spectra, df.log = 0.05, removeFirst = 1e+06, removeLast = 0,
  bLog = FALSE)
```

Arguments

`spectra`	spectra: list(spec,freq) spec[specIndex]: spectra density vector freq[specIndex]: frequency vector
`df.log`	width of the smoother in log units
`removeFirst`	elements to remove on the slow side (one element recommended because of the detrending
`removeLast`	elements to remove on the fast side
`bLog`	TRUE: average in the log space of the power, FALSE: arithmetic average

Value

smoothed spectrum

Author(s)

Thomas Laepple

Examples

```
x<-ts(arima.sim(list(ar = 0.9),1000))
spec<-SpecMTM(x)
LPlot(spec,col="grey")
LLines(LogSmooth(spec,df.log=0.01),lwd=2,col="green")
LLines(LogSmooth(spec,df.log=0.05),lwd=2,col="blue")
LLines(LogSmooth(spec,df.log=0.1),lwd=2,col="red")
legend("bottomleft",col=c("grey","green","blue","red"),lwd=2,c("raw","smoothed 0.01","smoothed 0.05","smoo
```

`Lowpass` *calculate weights for lowpass filter*

Description

based on Bloomfield 1976

Usage

```
Lowpass(omega.c, n = 9, sample = 1, convergence = T)
```

Arguments

`omega.c`	cutoff frequency
`n`	length of the filter, has to be odd
`sample`	sampling rate of the timeseries on which the filter will be applied
`convergence`	TRUE: smoothed least square lowpass; FALSE = unsmoothed

Value

vector of filter weights

Author(s)

Thomas Laepple

LPlot *add Logplot + transparent confidence interval for the spectral plotting*

Description

add Logplot + transparent confidence interval for the spectral plotting

Usage

```
LPlot(x, conf = TRUE, col = "black", alpha = 0.3, removeFirst = 0,
  removeLast = 0, xlab = "f", ylab = "PSD", ...)
```

Arguments

`x`	spectra object
`conf`	TRUE: Plot confidence interval
`col`	color
`alpha`	transparency
`removeFirst`	omit removeFirst values on the low frequency side
`removeLast`	omit removeFirst values on the high frequency side
`xlab`	label of x-axes
`ylab`	label of y-axes
`...`	other parameters to be passed to the line functio

Value

none

Author(s)

Thomas Laepple

Examples

```
x<-ts(arima.sim(list(ar = 0.9),1000))
spec<-SpecMTM(x)
LPlot(spec,col="grey")
LLines(LogSmooth(spec),lwd=2)
```

MakeEquidistant *Average an irregular timeseries to a regular timeseries*

Description

Average an irregular timeseries to a regular timeseries

Usage

```
MakeEquidistant(t.x, t.y, dt = 0.1, time.target = seq(from = t.x[1], to =
  t.x[length(t.x)], by = dt), dt.hres = NULL, bFilter = TRUE, k = 5,
  kf = 1.2)
```

Arguments

`t.x`	vector of timepoints
`t.y`	vector of corresponding values
`dt`	target timestep; can be omitted if time.target is supplied
`time.target`	time vector to which timeseries should be averaged/interpolated to by default the same range as t.x with a timestep dt
`dt.hres`	timestep of the intermediate high-resolution interpolation. Should be smaller than the smallest timestep
`bFilter`	(TRUE) low passs filter the data to avoid aliasing, (FALSE) just interpolate
`k`	scaling factor for the Length of the filter (increasing creates
`kf`	scaling factor for the lowpass frequency; 1 = Nyquist, 1.2 = 1.2xNyquist is a tradeoff between reducing variance loss and keeping aliasing small

Details

Make an irregular timeseries equidistant by interpolating to high resolution, lowpass filtering to the Nyquist frequency, and subsampling; e.g. as used in Huybers and Laepple, EPSL 2014

Value

ts object with the equidistant timeseries

Author(s)

Thomas Laepple

MeanSpectrum *average spectra with weighting*

Description

average spectra with weighting, spectra can have different resolution and span a different freq range

Usage

```
MeanSpectrum(specList, iRemoveLowest = 1, weights = rep(1,
  length(specList)))
```

Arguments

`iRemoveLowest`	number of lowest frequencies to remove (e.g. to remove detrending bias)
`weights`	vector of weights (same length as elements in speclist)
`speclist`	list of spectra

Details

Calculate the weighted mean spectrum of all spectra by interpolating them to the highest resolution

Value

list(spec,nRecords) spec=average spectrum, nRecords = number of records contributing to each spectral estimate

Author(s)

Thomas Laepple

SimPowerlaw *Simulate a random timeseries with a powerlaw spectrum*

Description

Simulate a random timeseries with a powerlaw spectrum

Usage

```
SimPowerlaw(beta, N)
```

Arguments

`beta`	slope
`N`	length of timeseries to be generated

Details

Method: FFT white noise, rescale, FFT back, the result is scaled to variance 1

Value

vector containing the timeseries

Author(s)

Thomas Laepple

`smoothlin.cutEnd` *smoothlin.cutEnd*

Usage

```
smoothlin.cutEnd(x, f, df.log, dof = 1)
```

Arguments

`x`

`f`

`df.log`

`dof`

Value

smoothed x

Author(s)

Thomas Laepple

`smoothlog` *smoothlog*

Usage

```
smoothlog(x, f, df.log)
```

Arguments

`x`

`f`

`df.log`

Value

smoothed x

Author(s)

Thomas Laepple

smoothlog.cutEnd *smoothlog.cutEnd*

Usage

```
smoothlog.cutEnd(x, f, df.log, dof = 1)
```

Arguments

`x`

`f`

`df.log`

`dof`

Value

smoothed x

Author(s)

Thomas Laepple

SpecInterpolate *Interpolates the spectrum spec to the specRef frequency resolution*

Usage

```
SpecInterpolate(freqRef, spec)
```

Arguments

`freqRef`	frequency vector of the target resolution
`spec`	list(spec,freq,dof)

Value

one spectum as list(spec,freq,dof) (spec on the specRef resolution)

Author(s)

Thomas Laepple

SpecMTM *MTM spectral estimator*

Description

MTM spectral estimator calls spec.mtm from library multitaper see spec.mtm from library mulitaper ?spec.mtm

Usage

```
SpecMTM(timeSeries, k = 3, nw = 2, nFFT = "default",
  centre = c("Slepian"), dpssIN = NULL, returnZeroFreq = FALSE,
  Ftest = FALSE, jackknife = FALSE, jkCIProb = 0.95,
  maxAdaptiveIterations = 100, plot = FALSE, na.action = na.fail,
  returnInternals = FALSE, detrend = TRUE, bPad = FALSE, ...)
```

Arguments

`timeSeries`	A time series of equally spaced data, this can be created by the ts() function where deltat is specified.
`k`	a positive integer, the number of tapers, often 2*nw.
`nw`	a positive double precision number, the time-bandwidth parameter.
`nFFT`	This function pads the data before computing the fft. nFFT indicates the total length of the data after padding.
`centre`	
`dpssIN`	
`returnZeroFreq`	
`Ftest`	
`jackknife`	
`jkCIProb`	
`maxAdaptiveIterations`	
`plot`	
`na.action`	
`returnInternals`	
`detrend`	
`bPad`	
`...`	

Value

spectra object list(freq,spec,dof) examples x<-ts(arima.sim(list(ar = 0.9),1000)) spec<-SpecMTM(x) LPlot(spec,col="grey") LLines(LogSmooth(spec),lwd=2)

Author(s)

Thomas Laepple

Index