Package: liftLRD (via r-universe)

September 13, 2024

Type Package

Title Wavelet Lifting Estimators of the Hurst Exponent for Regularly and Irregularly Sampled Time Series

Version 1.0-9

Date 2023-08-20

Author Marina Knight [aut], Guy Nason [ctb], Matt Nunes [aut, cre]

Maintainer Matt Nunes <nunesrpackages@gmail.com>

Description Implementations of Hurst exponent estimators based on the relationship between wavelet lifting scales and wavelet energy of Knight et al (2017) <doi:10.1007/s11222-016-9698-2>.

License GPL-2

Depends adlift, nlt

Suggests fracdiff

NeedsCompilation no

Repository CRAN

Date/Publication 2023-09-17 08:50:02 UTC

Contents

`liftLRD-package` *Wavelet lifting estimators of the Hurst exponent for regularly and irregularly sampled time series*

Description

Implementations of Hurst exponent estimators based on the relationship between wavelet lifting scales and wavelet energy

Details

This package exploits a wavelet transform for irregularly spaced data to form wavelet-like scale-based energy measures for a time series. This is then used to estimate the Hurst exponent. The main routine is

`liftHurst`

Author(s)

Marina Knight, Guy Nason, Matt Nunes

Maintainer: Matt Nunes <nunesrpackages@gmail.com>

References

Knight, M. I, Nason, G. P. and Nunes, M. A. (2017) A wavelet lifting approach to long-memory estimation. *Stat. Comput.* **27** (6), 1453–1471. DOI 10.1007/s11222-016-9698-2.

For related literature on the lifting methodology adopted in the technique, see

Nunes, M. A., Knight, M. I and Nason, G. P. (2006) Adaptive lifting for nonparametric regression. *Stat. Comput.* **16** (2), 143–159.

Knight, M. I. and Nason, G. P. (2009) A 'nondecimated' wavelet transform. *Stat. Comput.* **19** (1), 1–16.

For more information on long-memory processes, see e.g.

Beran, J. et al. (2013) Long-memory processes. Springer.

artificial.levels *artificial.levels*

Description

This function splits the coefficients into levels according to either (i) increasing quantiles of the removed interval lengths or (ii) dyadic splitting relative to a fixed lowest scale

Usage

```
artificial.levels(y, rem, time, tail = TRUE, type = 1)
```

Arguments

`y`	a vector of the removed interval lengths (in the order of `removelist`).
`rem`	vector of indices of the removed points (from the output of the forward transform).
`time`	vector of observed times for the decomposed signal.
`tail`	a boolean variable indicating whether coarse levels with a small number of detail coefficients (less than 10 coefficients) should be combined.
`type`	an integer indicating which type of artificial levels to compute. If `type==1`, the "usual" artificial levels using quantiles of the scales; otherwise the levels are computed using quantiles being a multiplicative factor of a fixed number. See details for more information.

Details

The function computes the so-called artificial levels of a set of removed integrals and corresponding detail coefficients, to mimic the dyadic level splitting in a classical wavelet framework. Details on the "usual" quantile-based splitting can be found in `artlev`. If `type==2` or `type==3`, the artificial levels are defined by intervals of the form [a0 2^j,a0 2^(j-1)) as described in Jansen et al. (2009), with a0 = 0.5 for `type==2` and set to the minimum sampling interval for `type==3`. The amalgamation of coarser artificial levels prevents variable energies at coarser scales affecting the predicted relationship between the wavelet scales and their corresponding energies.

Value

`p`	a list of the grouped indices of `removelist` (in decreasing group size) indicating thresholding groups.

Author(s)

Matt Nunes, Marina Knight

References

Jansen, M, Nason, G. P. and Silverman, B. W. (2009) Multiscale methods for data on graphs and irregular multidimensional situations. *J. Roy. Stat. Soc. B* **71**, Part 1, 97–125.

See Also

liftHurst

Examples

```
#create test signal data
#
library(adlift)
x<-runif(100)
y<-make.signal2("blocks",x=x)
#
#perform forward transform...
#
out<-fwtnp(x,y,LocalPred=AdaptNeigh,neighbours=2)
#
al<-artificial.levels(out$lengthsremove,out$removelist, x, type = 1)
#
#
# the indices of removelist split into levels:
al
#
```

---

bootci *bootstrap confidence interval calculation*

Description

This function uses the Hurst exponent estimates from different lifting trajectories to form a bootstrap confidence interval

Usage

```
bootci(x, level = 0.05)
```

Arguments

x	a vector of estimates for the Hurst exponent.
level	the significance level for which to compute the bootstrap confidence interval.

Value

A vector of length 2, indicating the lower and upper confidence interval values.

Author(s)

Matt Nunes

References

Knight, M. I, Nason, G. P. and Nunes, M. A. (2017) A wavelet lifting approach to long-memory estimation. *Stat. Comput.* **27** (6), 1453–1471. DOI 10.1007/s11222-016-9698-2.

See Also

liftHurst

Examples

```
x<-rnorm(100,0.7,0.24) # vector representing Hurst estimates from NLT

bootci(x)
```

Hfrombeta — *Compute Hurst exponent from wavelet scale - energy regression slope*

Description

Uses the slope of the relationship between wavelet scale and wavelet energy to compute an estimate of the Hurst exponent

Usage

```
Hfrombeta(beta, model = c("FBM","FGN","ID"))
```

Arguments

beta	The estimated slope of the relationship between wavelet scale and energy.
model	The assumed long-range dependence model for the time series under analysis.

Details

There is a theoretical linear relationship growth in the (log) wavelet energy for increasing wavelet scale. This corresponds to the decay in the autocorrelation of a (long range dependent) time series being analysed, and therefore the Hurst exponent, H. The specific relation to H is dependent to the assumed model; in particular for a Fractional Brownian motion, the relationship between H and the slope is H = abs(beta - 1)/2, whereas for Fractional Gaussian noise or dth order Fractional differenced series, the relationship is H = (beta+1)/2.

Value

H	The Hurst exponent, computed for a specific beta and underlying model.

Author(s)

Matt Nunes

References

Knight, M. I, Nason, G. P. and Nunes, M. A. (2017) A wavelet lifting approach to long-memory estimation. *Stat. Comput.* **27** (6), 1453–1471. DOI 10.1007/s11222-016-9698-2.

Beran, J. et al. (2013) Long-Memory Processes. Springer.

See Also

liftHurst

Examples

```
Hfrombeta(0.8,model="FGN")
```

---

idj *Functions to perform summary calculations of wavelet scales and energies.*

Description

To estimate the slope of the relationship between wavelet scale and wavelet energy, choices have to be made as to how these quantities are computed. Examples of these choices are the functions listed here.

Usage

```
idj(x, j)
meanj(x, j)
medj(x, j)
mean2(x)
mad2(x)
meanmo(x)
```

Arguments

x	a vector of values, representing the integrals or detail coefficients in a particular artificial level
j	the index of the artificial level.

Value

A numeric value corresponding to the average squared detail coefficient, squared mean absolute deviation, median scale etc.

Author(s)

Matt Nunes

References

Knight, M. I, Nason, G. P. and Nunes, M. A. (2017) A wavelet lifting approach to long-memory estimation. *Stat. Comput.* **27** (6), 1453–1471. DOI 10.1007/s11222-016-9698-2.

See Also

`liftHurst`

Examples

```
x<-rnorm(50,30,2)

# calculate the average squared value of x (i.e. energy)

mean2(x)
```

`liftHurst` *Performs (non-decimated) lifting based estimation of the Hurst exponent*

Description

The function exploits the linear relationship in wavelet energy per scale to estimate the long range dependence parameter of a irregular time series.

Usage

```
liftHurst(x, grid = 1:length(x), model = "FGN", ntraj = 50,
tradonly = FALSE, cutoffs = 0, cut.fine = TRUE, efun = meanmo,
afun = idj, altype = 1, tail = TRUE, normalise = TRUE,
level = 0.05, bc = TRUE, vc = TRUE, jsc = TRUE, BHonly=TRUE,
verbose = FALSE, ...)
```

Arguments

x — A time series, possibly irregularly spaced.

grid — The time samples corresponding to the time series x.

model — The underlying model the time series is assumed to follow. Possible values are "FBM", "FGN", and "ID".

ntraj — The number of lifting trajectories (bootstraps) used for the estimation of the Hurst exponent.

tradonly — A boolean variable, indicating whether the "traditional" minimum integral lifting trajectory should be used for the estimation (tradonly=TRUE). In this case, the argument ntraj is not used.

cutoffs — A vector indicating if the coarsest scales are to be removed when estimating the linear slope in the energy-scale relationship; for example if cutoffs = 0, all levels will be used in the estimation. For multiple cutoffs, all slopes will be calculated.

cut.fine — A boolean variable indicating if the finest (default) or coarsest scales are to be removed when estimating the linear slope in the energy-scale relationship; see cutoffs description for more information.

efun — A function indicating which measure of wavelet energy to use. The default is meanmo, corresponding to the average squared detail coefficient within a particular scale, scaled by (n-1). Other measures could be mad2 (squared median absolute deviation of the detail coefficients), but the average squared detail coefficient has been shown to be more reliable than other measures.

afun — A function indicating which measure of wavelet integral (scale) to use (corresponding to the x-axis of the energy-scale relationship). The default is idj, corresponding to the log2 of the dyadic wavelet scale, i.e an integer representing the scale. Other measures could be meanj or medj, representing the average (respectively median) integral value for each artificial scale.

altype — An integer indicating which type of artificial levels to be used, see artificial.levels

for more information.

tail — A boolean variable indicating whether to amalgamate artificial levels with few coefficients, see artificial.levels

for more information.

normalise — A boolean variable indicating whether to normalise the detail coefficients by their individual (induced) standard deviations (computed using the diagonal of the lifting matrix).

level — The significance level for the bootstrap confidence interval of the Hurst exponent estimate.

bc — A boolean variable indicating whether bias-correction should take place or not, using the approximate chi^2 distribution of the wavelet energies.

`vc`	A boolean variable indicating whether a weighted linear regression should be used when estimating the Hurst exponent. If `TRUE`, the inverse of the variances of the approximate chi^2 distribution of the wavelet energies are used in the regression.
`jsc`	A boolean variable indicating whether the slope of the log-linear relationship between the artificial scales and the log of the integrals should be computed and used to reweight the estimate of the Hurst exponent.
`BHonly`	A boolean variable indicating whether only the Hurst estimate should be returned. If `FALSE`, then the (log) energies used in the calculation is also returned. Note that if bootstrapping is performed (`tradonly=FALSE`), then the energies for the last trajectory will be returned.
`verbose`	A boolean variable indicating whether informative status messages should be printed to the screen.
`...`	Any other optional arguments to the function `fwtnpperm` function from the `nlt` package, which performs wavelet lifting of the time series `x` according to a specified lifting trajectory.

Details

Wavelet lifting is performed on a time series to convert it into a set of wavelet coefficients and corresponding lifting integrals, specific to when the data were "lifted" during the decomposition. The coefficients are then grouped into artificial levels, using the integrals to mimic the support of the wavelets in the classical wavelet setting, and therefore producing a notion of scale. The coefficients in each artificial level are then used to compute values of the wavelet energy for a particular level. The (slope of the) linear relationship between the scales and their energies is then used in computing an estimate of the Hurst exponent for the series. This procedure can be performed for multiple (random) lifting trajectories, each producing a slightly different estimate.

Value

If `tradonly=TRUE`, the function returns a matrix of dimension `length(cutoffs)` x 2. The first column are the slopes of the regression fits for each cutoff, whereas the second column are the corresponding estimates for the Hurst exponent.

If `tradonly=FALSE`, the function returns a matrix of dimension `length(cutoffs)` x 5. The first column are the slopes of the regression fits for each cutoff, where the average is taken over the `ntraj` randomly generated lifting trajectories. Similarly, the second column represents the average Hurst exponent for the cutoffs over all lifting paths. The third column is the standard deviation of the `ntraj` Hurst estimates through performing non-decimated lifting. The fourth and fifth columns are the lower and upper values of the bootstrap confidence interval of the Hurst exponent estimate.

If `BHonly=FALSE`, the routine also returns the energies and scales (on a log scale) which are used in the regression to estimate the decay properties of the spectrum (for the last lifting trajectory), as well as the weights used in the regression (if `vc = TRUE`). if `jsc = TRUE`, the slope of the integral log-linear relationship is also returned.

Author(s)

Marina Knight, Matt Nunes

References

Knight, M. I, Nason, G. P. and Nunes, M. A. (2017) A wavelet lifting approach to long-memory estimation. *Stat. Comput.* **27** (6), 1453–1471. DOI 10.1007/s11222-016-9698-2.

For more details on the weighted linear regression and bias calculations, see e.g.

Veitch, D. and Abry, P. (1999) A Wavelet-Based Joint Estimator of the Parameters of Long-Range Dependence. *IEEE Trans. Info. Theory* **45** (3), 878–897.

See Also

`artificial.levels`, `Hfrombeta`, `fwtnpperm`

Examples

```
library(fracdiff)

# simulate a long range dependent time series
x<-fracdiff.sim(n = 200, d = 0.3)$series

# perform lifting-based estimation of the Hurst exponent

Hestx<-liftHurst(x, tradonly=TRUE)

# An example with missingness
x1<-fracdiff.sim(n = 500, d = 0.3)$series
gap1<-30:40
gap2<-77:90
gap3<-146:166
timeindex<-setdiff(1:500,c(gap1,gap2,gap3))

Hestx1<-liftHurst(x1[timeindex],grid=timeindex, tradonly=TRUE)
```

Index