Package ‘haarfisz’

February 20, 2015

Version 4.5

Date 25/11/2009

Title Software to perform Haar Fisz transforms

Author Piotr Fryzlewicz <P.Fryzlewicz@lse.ac.uk>

Maintainer Guy Nason <G.P.Nason@bristol.ac.uk>

Depends R (>= 2.0), wavethresh

Description A Haar-Fisz algorithm for Poisson intensity estimation

License GPL (>= 2)

Repository CRAN

Date/Publication 2010-03-16 17:23:50

NeedsCompilation no

R topics documented:

denoise.poisson *denoise.poisson*

Description

Main routine of the package. Estimates the deterministic discretised intensity of a one-dimensional Poisson process using the Haar-Fisz transformation and partial cycle spinning. Requires WaveThresh3.

Usage

```
denoise.poisson(y, meth.1 = hf.bt, cs.1 = 50, meth.2 = hf.cv, cs.2 = 50, hybrid = TRUE)
```

Arguments

`y`	The vector of Poisson counts, its length must be a power of 2.
`meth.1`	Unquoted name of an S-Plus routine for denoising Gaussian contaminated vectors. Must take and return a vector of length 2^J where J is an integer. The following routines supplied in this package can be used here: `hf.u`, `hf.cv`, `hf.bt`, `hf.tiu`. The user can define and plug in his or her own routines here.
`cs.1`	The number of cycle spins to be performed with `meth.1`. Must be between 1 and N-1, where N is the length of `y`.
`meth.2`	Of the same type as `meth.1`.
`cs.2`	The number of cycle spins to be performed with `meth.2`.
`hybrid`	If set to TRUE, then the estimates are computed using both `meth.1` with `cs.1` cycle spins, and `meth.2` with `cs.2` cycle spins, and the final estimate is taken to be the average of these two. If set to `FALSE`, only `meth.1` with `cs.1` cycle spins is used to compute the final estimate.

Value

Returns:

`est`	The estimate of the intensity which underlies `y` (a vector of the same length as `y`).

Author(s)

Piotr Fryzlewicz

`hf.bt` *hf.bt*

Description

Denoises a Gaussian contaminated vector using a version of the wavelet-based “greedy tree" algorithm by Baraniuk, see the reference in the thesis.

Usage

```
hf.bt(x, filter.number = 1, family = "DaubExPhase", min.level = 3, noise.level = NULL)
```

Arguments

`x`	The noisy vector, its length must be a power of 2.
`filter.number`	The filter number of the analysing wavelet. Can be set to 1, 2, ..., 10 for `family == "DaubExPhase"`, or to 4, 5, ..., 10 for `family == "DaubLeAsymm"`.
`family`	The family of wavelet bases from which the wavelet `filter.number` is chosen. Can be set to "DaubExPhase" or "DaubLeAsymm".
`min.level`	The minimum level thresholded.
`noise.level`	Standard deviation of the noise, can be set to a positive number or NULL; in the latter case it will be estimated using MAD.

Value

Returns: itemestDenoised version of `x`.

Author(s)

Piotr Fryzlewicz

`hf.cv` *hf.cv*

Description

Denoises a Gaussian contaminated vector using wavelet thresholding with a threshold chosen by “leave-half-out" cross-validation. Requires WaveThresh3. Also see help to wd, threshold and wr in WaveThresh.

Usage

```
hf.cv(x, filter.number = 1, family = "DaubExPhase", min.level = 3, type = "hard")
```

Arguments

`x`	The noisy vector, its length must be a power of 2.
`filter.number`	The filter number of the analysing wavelet. Can be set to 1, 2, ..., 10 for `family == "DaubExPhase"`, or to 4, 5, ..., 10 for `family == "DaubLeAsymm"`.
`family`	The family of wavelet bases from which the wavelet `filter.number` is chosen. Can be set to "DaubExPhase" or "DaubLeAsymm".
`min.level`	The minimum level thresholded.
`type`	Type of thresholding, can be set to "hard" or "soft".

Value

Returns:

`est`	Denoised version of `x`

Author(s)

Piotr Fryzlewicz

`hf.tiu` *hf.tiu*

Description

Denoises a Gaussian contaminated vector using translation-invariant hard wavelet thresholding with the universal threshold. Requires WaveThresh3. Also see help to `wd`, `threshold`, `convert` and `AvBasis` in WaveThresh.

Usage

```
hf.tiu(x, filter.number = 1, family = "DaubExPhase", min.level = 3, noise.level = 1)
```

Arguments

`x`	The noisy vector, its length must be a power of 2.
`filter.number`	The filter number of the analysing wavelet. Can be set to 1, 2, ..., 10 for `family == "DaubExPhase"`, or to 4, 5, ..., 10 for `family == "DaubLeAsymm"`.
`family`	The family of wavelet bases from which the wavelet `filter.number` is chosen. Can be set to "DaubExPhase" or "DaubLeAsymm".
`min.level`	The minimum level thresholded.
`noise.level`	Standard deviation of the noise, can be set to a positive number or to an estimate (a function of x).

Value

Returns:

`est`	Denoised version of `x`.

Author(s)

Piotr Fryzlewicz

`hf.u` *hf.u*

Description

Denoises a Gaussian contaminated vector using wavelet thresholding with the universal threshold. Requires WaveThresh3. Also see help to `wd`, `threshold`,and `link{wr}` in WaveThresh.

Usage

```
hf.u(x, filter.number = 10, family = "DaubLeAsymm", min.level = 3, type = "hard")
```

Arguments

`x`	The noisy vector, its length must be a power of 2.
`filter.number`	The filter number of the analysing wavelet. Can be set to 1, 2, ..., 10 for `family == "DaubExPhase"`, or to 4, 5, ..., 10 for `family == "DaubLeAsymm"`.
`family`	The family of wavelet bases from which the wavelet `filter.number` is chosen. Can be set to "DaubExPhase" or "DaubLeAsymm".
`min.level`	The minimum level thresholded.
`type`	Type of thresholding, can be set to hard or soft

Value

Returns:

`est`	Denoised version of `x`.

Author(s)

Piotr Fryzlewicz

hft *hft*

Description

Performs the Haar-Fisz transform.

Usage

```
hft(data)
```

Arguments

`data`	The vector of Poisson counts, its length must be a power of 2

Details

The inverse transform is `hft.inv`

Value

Returns:

`hfy`	The Haar-Fisz transform of codedata (vector of the same length as `data`).

Author(s)

Piotr Fryzlewicz

See Also

`denoise.poisson`, `hft.inv`, `hf.bt`, `hf.cv`, `hf.u`, `hf.tiu`

Examples

```
#
# Generate Poisson data, half with one intensity, and half with a larger one
#
v <- c( rpois(64, lambda=1), rpois(64, lambda=10))
#
# Plot it to note that the variation is bigger in the second half
# (and the mean, but this is not important for this bit)
#
## Not run: ts.plot(v)
#
# Now do the Haar-Fisz transform
#
vhft <- hft(v)
#
# Now plot this, and see that the variance of the second bit is now comparable
# to the first
#
## Not run: ts.plot(vhft)
```

`hft.inv` *hft.inv*

Description

Performs the inverse Haar-Fisz transform.

Usage

```
hft.inv(data)
```

Arguments

`data`	Vector of length 2^J where J is an integer

Value

Returns:

`ihfx`	The inverse Haar-Fisz transform of `x` (vector of the same length as `data`).

Author(s)

Piotr Fryzlewicz

`shift.sequence` *shift.sequence*

Description

One of my functions to resolve issues for a similar function that seems to have been forgotten in haarfisz.

Usage

```
shift.sequence(v, places, dir="right")
```

Arguments

`v`	Vector to shift
`places`	The number of places to shift
`dir`	Whether the shift should be right or left

Details

This function takes a sequence input and shifs it to the left or right by the specified number of places.

Value

a shifted output sequence.

Author(s)

Piotr Fryzlewicz

`xquake` *xquake*

Description

Time series of the number of earthquakes of magnitude >= 3.0 which occurred in Northern California in 1024 weeks, the last week being 29/11 – 05/12/2000.

Usage

```
data(xquake)
```

Source

The series was composed using the data obtained from the Northern California Earthquake Data Center.

Index