

# Package ‘haarfisz’

July 22, 2025

**Type** Package

**Title** Software to Perform Haar Fisz Transforms

**Version** 4.5.4

**Date** 2023-09-01

**Depends** R (>= 3.5.0), wavethresh

**Description** A Haar-Fisz algorithm for Poisson intensity estimation.

Will denoise Poisson distributed sequences where underlying intensity is not constant. Uses the multiscale variance-stabilization method called the Haar-Fisz transform. Contains functions to carry out the forward and inverse Haar-Fisz transform and denoising on near-Gaussian sequences. Can also carry out cycle-spinning.

Main reference: Fryzlewicz, P. and Nason, G.P. (2004) "A Haar-Fisz algorithm for Poisson intensity estimation." Journal of Computational and Graphical Statistics, 13, 621-638. <doi:10.1198/106186004X2697>.

**License** GPL (>= 2)

**NeedsCompilation** no

**Author** Piotr Fryzlewicz [aut],  
Guy Nason [cre]

**Maintainer** Guy Nason <g.nason@imperial.ac.uk>

**Repository** CRAN

**Date/Publication** 2023-09-01 11:20:02 UTC

## Contents

haarfisz-package	2
denoise.poisson	3
hf.bt	4
hf.cv	5
hf.tiu	7
hf.u	8

hft . . . . .	9
hft.inv . . . . .	11
shift.sequence . . . . .	12
xquake . . . . .	13
<b>Index</b>	<b>14</b>

---

haarfisiz-package	<i>A Haar-Fisz Algorithm for Poisson Intensity Estimation.</i>
-------------------	--

---

## Description

Package to denoise Poisson distributed sequence where underlying intensity is not constant. Uses the multiscale variance-stabilization method called the Haar-Fisz transform. Contains functions to carry out the forward and inverse Haar-Fisz transform and denoising on near-Gaussian sequences. Can also carry out cycle-spinning.

## Details

Package to denoise Poisson distributed sequence where underlying intensity is not constant. Uses the multiscale variance-stabilization method called the Haar-Fisz transform. Contains functions to carry out the forward and inverse Haar-Fisz transform and denoising on near-Gaussian sequences. Can also carry out cycle-spinning. See main routine [denoise.poisson](#)

## Author(s)

Piotr Fryzlewicz>

## References

- Fryzlewicz, P. (2003) Wavelet Techniques for Time Series and Poisson Data. *PhD Thesis*, University of Bristol, Bristol, UK <https://www.ma.imperial.ac.uk/~gnason/Research/MAPZFthesis.ps.gz>
- Fryzlewicz, P. and Nason, G.P. (2004) A Haar-Fisz algorithm for Poisson intensity estimation. *Journal of Computational and Graphical Statistics*, **13**, 621-638. doi:10.1198/106186004X2697
- Nason, G.P. (2008) *Wavelet Methods in Statistics with R*. Springer: New York (Section 6.4) doi:10.1007/9780387759616

## See Also

[denoise.poisson](#), [hft](#), [hft.inv](#)

**Examples**

```
#
#
# Main Poisson denoising function is denoise.poisson
#
# Forward Haar-Fisz transform is hft
#
# Inverse Haar-Fisz transform is hft.inv
```

---

denoise.poisson	<i>denoise.poisson</i>
-----------------	------------------------

---

**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.

**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: <a href="#">hf.u</a> , <a href="#">hf.cv</a> , <a href="#">hf.bt</a> , <a href="#">hf.tiu</a> . 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.

**Details**

For a given input sequence, basic operation of the code performs a cyclic shift on the data. Then applies the Haar-Fisz transform, then one of the denoising methods (specified by meth.1), then the inverse Haar-Fisz transform and then a shift back. This is repeated for cs.1 cyclic shifts and the results of all shifts returned.

**Value**

Returns vector of the same length as the input y but is the denoised estimate.

**Author(s)**

Piotr Fryzlewicz

**References**

Fryzlewicz, P. and Nason, G.P. (2004) A Haar-Fisz algorithm for Poisson intensity estimation. *Journal of Computational and Graphical Statistics*, **13**, 621-638. doi:[10.1198/106186004X2697](https://doi.org/10.1198/106186004X2697)

**See Also**

[hft](#), [hft.inv](#), [hf.u](#), [hf.cv](#), [hf.bt](#), [hf.tiu](#)

**Examples**

```
#
# Apply denoise.poisson to xquake data
#
data(xquake)
xquake.denoised <- denoise.poisson(xquake)
#
# Now plot the original data and it's denoised version in red
#
plot(xquake, type="l")
lines(xquake.denoised, col=2)
```

---

hf.bt

*hf.bt*


---

**Description**

Denoises a Gaussian contaminated vector using a version of the wavelet-based “greedy tree” algorithm by Baraniuk, see references. Note: this function does not actually do any Haar-Fisz transform, it is a homogeneous variance Gaussian denoising function, which is used by the **haarfisz** package.

**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**

Denoised version of  $x$ .

**Author(s)**

Piotr Fryzlewicz

**References**

Baraniuk, R. G. (1999) Optimal tree approximation with wavelets. Pages 206-214 of Unser, M.A., Aldroubi, A. and Laine, A.F. (eds), *Wavelet Applications in Signal and Image Processing VII*, Proceedings of SPIE **3813**. SPIE. doi:[10.1117/12.366780](https://doi.org/10.1117/12.366780)

Fryzlewicz, P. and Nason, G.P. (2004) A Haar-Fisz algorithm for Poisson intensity estimation. *Journal of Computational and Graphical Statistics*, **13**, 621-638. doi:[10.1198/106186004X2697](https://doi.org/10.1198/106186004X2697)

**Examples**

```
#
# Generate simple sinusoidal test signal
#
test.sig <- sin(seq(from=0, to=6*pi, length=128))
#
# Invent simulated noisy signal
#
test.dat <- test.sig + rnorm(128, sd=0.2)
#
# Denoise using hf.bt
#
test.est <- hf.bt(test.dat)
#
# Now plot the results: the truth, the noisy signal, the estimate
#
ts.plot(test.dat)
lines(test.est, col=2)
lines(test.sig, col=3, lty=2)
```

---

hf.cv

*hf.cv*

---

**Description**

Denoises a Gaussian contaminated vector using wavelet thresholding with a threshold chosen by "leave-half-out" cross-validation. Note: this function does not actually do any Haar-Fisz transform, it is a homogeneous variance Gaussian denoising function, which is used by the **haarfisz** package.

**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".

**Details**

Uses [threshold](#), [wd](#) and [AvBasis](#)

**Value**

Denoised version of x

**Author(s)**

Piotr Fryzlewicz

**References**

Fryzlewicz, P. and Nason, G.P. (2004) A Haar-Fisz algorithm for Poisson intensity estimation. *Journal of Computational and Graphical Statistics*, **13**, 621-638. doi:[10.1198/106186004X2697](https://doi.org/10.1198/106186004X2697)

**See Also**

[threshold](#), [wd](#), [AvBasis](#)

**Examples**

```
#
# Generate simple sinusoidal test signal
#
test.sig <- sin(seq(from=0, to=6*pi, length=128))
#
# Invent simulated noisy signal
#
test.dat <- test.sig + rnorm(128, sd=0.2)
#
# Denoise using hf.bt
#
test.est <- hf.cv(test.dat)
#
# Now plot the results: the truth, the noisy signal, the estimate
#
ts.plot(test.dat)
lines(test.est, col=2)
lines(test.sig, col=3, lty=2)
```

---

`hf.tiu`*hf.tiu*

---

**Description**

Denoises a Gaussian contaminated vector using translation-invariant hard wavelet thresholding with the universal threshold. Note: this function does not actually do any Haar-Fisz transform, it is a homogeneous variance Gaussian denoising function, which is used by the **haarfisz** package.

**Usage**

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

**Arguments**

<code>x</code>	The noisy vector, its length must be a power of 2.
<code>filter.number</code>	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".
<code>family</code>	The family of wavelet bases from which the wavelet <code>filter.number</code> is chosen. Can be set to "DaubExPhase" or "DaubLeAsymm".
<code>min.level</code>	The minimum level thresholded.
<code>noise.level</code>	Standard deviation of the noise, can be set to a positive number or to an estimate (a function of <code>x</code> ).

**Details**

Uses [threshold](#), [wd](#) and [AvBasis](#)

**Value**

Denoised version of `x`.

**Author(s)**

Piotr Fryzlewicz

**References**

Fryzlewicz, P. and Nason, G.P. (2004) A Haar-Fisz algorithm for Poisson intensity estimation. *Journal of Computational and Graphical Statistics*, **13**, 621-638. doi:10.1198/106186004X2697

**See Also**

[threshold](#), [wd](#), [AvBasis](#)

**Examples**

```

#
# Generate simple sinusoidal test signal
#
test.sig <- sin(seq(from=0, to=6*pi, length=128))
#
# Invent simulated noisy signal
#
test.dat <- test.sig + rnorm(128, sd=0.2)
#
# Denoise using hf.bt
#
test.est <- hf.tiu(test.dat)
#
# Now plot the results: the truth, the noisy signal, the estimate
#
ts.plot(test.dat)
lines(test.est, col=2)
lines(test.sig, col=3, lty=2)

```

---

hf.u

*hf.u*


---

**Description**

Denoises a Gaussian contaminated vector using wavelet thresholding with the universal threshold. Note: this function does not actually do any Haar-Fisz transform, it is a homogeneous variance Gaussian denoising function, which is used by the **haarfisz** package.

**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

**Details**

Uses [threshold](#), [wd](#) and [AvBasis](#)



**Value**

Denoised version of  $x$ .

**Author(s)**

Piotr Fryzlewicz

**References**

Fryzlewicz, P. and Nason, G.P. (2004) A Haar-Fisz algorithm for Poisson intensity estimation. *Journal of Computational and Graphical Statistics*, **13**, 621-638. doi:[10.1198/106186004X2697](https://doi.org/10.1198/106186004X2697)

**See Also**

[threshold](#), [wd](#), [AvBasis](#)

**Examples**

```
#
# Generate simple sinusoidal test signal
#
test.sig <- sin(seq(from=0, to=6*pi, length=128))
#
# Invent simulated noisy signal
#
test.dat <- test.sig + rnorm(128, sd=0.2)
#
# Denoise using hf.bt
#
test.est <- hf.u(test.dat)
#
# Now plot the results: the truth, the noisy signal, the estimate
#
ts.plot(test.dat)
lines(test.est, col=2)
lines(test.sig, col=3, lty=2)
```

---

hft

*hft*

---

**Description**

Performs the (forward) Haar-Fisz transform.

**Usage**

```
hft(data)
```

**Arguments**

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

**Details**

The Haar-Fisz for Poisson works, roughly speaking, by taking the Haar wavelet transform of data. Then dividing the mother wavelet coefficients by the respective father coefficients, and replacing the results of the divisions back into the same coefficient locations, and then carrying out an inverse Haar wavelet transform. This produces a nearer-Gaussian variance stabilized version of the original (or a version of the underlying intensity which is close to an ‘intensity PLUS homogeneous Gaussian noise’ signal, which is easier to denoise using ‘standard’ methods.

The inverse transform is [hft.inv](#)

**Value**

The Haar-Fisz transform of data, which will be the same length as data.

**Author(s)**

Piotr Fryzlewicz

**References**

Fryzlewicz, P. and Nason, G.P. (2004) A Haar-Fisz algorithm for Poisson intensity estimation. *Journal of Computational and Graphical Statistics*, **13**, 621-638. doi:[10.1198/106186004X2697](https://doi.org/10.1198/106186004X2697)

**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)
#
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
#
ts.plot(vhft)
```

---

hft.inv	<i>hft.inv</i>
---------	----------------

---

**Description**

Performs the inverse Haar-Fisz transform.

**Usage**

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

**Arguments**

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

**Value**

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

**Author(s)**

Piotr Fryzlewicz

**References**

Fryzlewicz, P. and Nason, G.P. (2004) A Haar-Fisz algorithm for Poisson intensity estimation. *Journal of Computational and Graphical Statistics*, **13**, 621-638. doi:[10.1198/106186004X2697](https://doi.org/10.1198/106186004X2697)

**See Also**

[denoise.poisson](#), [hft](#)

**Examples**

```
#
# Make up test set (mimics sequence with half low intensity, followed by
# half high intensity)
#
test.set <- c(8,5,6,3, 30,40,20,35)
#
# Do forward HFT
#
test.hft <- hft(test.set)
test.hft
# [1] 16.38621 15.20951 15.65216 14.23795 21.20421 22.89452 19.27753 22.13791
#
# Do inverse HFT
#
test.back <- hft.inv(test.hft)
```

```
test.back
# [1] 8 5 6 3 30 40 20 35
#
# Same as original
#
```

---

shift.sequence	<i>shift.sequence</i>
----------------	-----------------------

---

### 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 shifts it to the left or right by the specified number of places. This is a circular shift. For example, when shifting to the right, any numbers that drop off are appended circularly to the front, etc.

### Value

a shifted output sequence.

### Author(s)

Piotr Fryzlewicz

### References

Fryzlewicz, P. and Nason, G.P. (2004) A Haar-Fisz algorithm for Poisson intensity estimation. *Journal of Computational and Graphical Statistics*, **13**, 621-638. doi:[10.1198/106186004X2697](https://doi.org/10.1198/106186004X2697)

**Examples**

```
#
# Shift 1:10 one place to the right
#
shift.sequence(1:10,1, dir="right")
# [1] 10 1 2 3 4 5 6 7 8 9

#
# Shift 1:10 twos place to the right
#
shift.sequence(1:10,2, dir="right")
# [1] 9 10 1 2 3 4 5 6 7 8

#
# Shift 1:10 one place to the left
#
shift.sequence(1:10,1, dir="left")
# [1] 2 3 4 5 6 7 8 9 10 1
```

---

xquake

*xquake*

---

**Description**

Time series of the number of earthquakes of magnitude  $\geq 3.0$  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

## \* datasets

xquake, [13](#)

## \* manip

denoise.poisson, [3](#)

hf.bt, [4](#)

hf.cv, [5](#)

hf.tiu, [7](#)

hf.u, [8](#)

hft, [9](#)

hft.inv, [11](#)

shift.sequence, [12](#)

## \* math

haarfisz-package, [2](#)

AvBasis, [6–9](#)

denoise.poisson, [2, 3, 10, 11](#)

haarfisz (haarfisz-package), [2](#)

haarfisz-package, [2](#)

hf.bt, [3, 4, 4, 10](#)

hf.cv, [3, 4, 5, 10](#)

hf.tiu, [3, 4, 7, 10](#)

hf.u, [3, 4, 8, 10](#)

hft, [2, 4, 9, 11](#)

hft.inv, [2, 4, 10, 11](#)

shift.sequence, [12](#)

threshold, [6–9](#)

wd, [6–9](#)

xquake, [13](#)