

# Package: binhf (via r-universe)

September 11, 2024

**Title** Haar-Fisz Functions for Binomial Data

**Version** 1.0-3

**Date** 2018-07-18

**Author** Matt Nunes <nunesrpackages@gmail.com>

**Depends** R (>= 2.10), wavethresh, adlift (>= 0.9.2), EbayesThresh

**Description** Binomial Haar-Fisz transforms for Gaussianization as in Nunes and Nason (2009).

**Maintainer** Matt Nunes <nunesrpackages@gmail.com>

**License** GPL (>= 2)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2018-07-18 22:40:12 UTC

## Contents

afgen . . . . .	2
ansc . . . . .	3
asymean . . . . .	4
asyvar . . . . .	5
binhf.wd . . . . .	6
Blocks . . . . .	7
chr20 . . . . .	8
ebayesthresh.wavelet.wd . . . . .	8
free . . . . .	9
freeinv . . . . .	10
hf.inv2 . . . . .	11
hfdnoise . . . . .	11
hfdnoise.wav . . . . .	13
ht . . . . .	14
ht.inv . . . . .	15
invansc . . . . .	16
invbinhf.wd . . . . .	17

norm . . . . .	18
pintens . . . . .	19
plotest . . . . .	19
propest.wav . . . . .	20
qqnormy . . . . .	22
qqstuff . . . . .	22
shift . . . . .	24
simsij . . . . .	25
statgen . . . . .	26

<b>Index</b>	<b>28</b>
--------------	-----------

---

afgen	<i>NN and Anscombe samples</i>
-------	--------------------------------

---

## Description

Samples binomial Fisz and Anscombe transformed random variables on a grid of binomial probabilities.

## Usage

```
afgen(xgrid = seq(0, 1, length = 21), ygrid = seq(0, 1, length = 21), samples = 1000,
      binsize = 32)
```

## Arguments

xgrid	vector of x co-ordinate probabilities.
ygrid	vector of x co-ordinate probabilities.
samples	the number of samples to draw from each random variable.
binsize	the binomial size of the binomial random variables.

## Details

The function produces sampled values from the random variable:

$$\zeta(X_1, X_2) = \frac{X_1 - X_2}{\sqrt{(X_1 + X_2)(2 * binsize - X_1 - X_2) / 2 * binsize}},$$

where  $X_i$  are  $\text{Bin}(\text{binsize}, p_i)$  random variables, for all combinations of values of  $p_1$  in xgrid and  $p_2$  in ygrid. For Anscombe's transformation,  $A = \sin^{-1} \sqrt{(x + 3/8) / (\text{binsize} + 3/4)}$ , the values correspond to the random variable with the larger binomial probability.

## Value

a	an array of dimensions $\text{length}(\text{xgrid}) \times \text{length}(\text{ygrid}) \times \text{samples}$ of values of binomial Haar-Fisz random variable.
b	an array of dimensions $\text{length}(\text{xgrid}) \times \text{length}(\text{ygrid}) \times \text{samples}$ of values of A.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**References**

Anscombe, F.J. (1948) The transformation of poisson, binomial and negative binomial Data, *Biometrika*, **35**, 246–254.

Nunes, M. and Nason, G.P. (2009) A multiscale variance stabilization for binomial sequence proportion estimation. *Statistica Sinica*, **19** (1491–1510).

**See Also**

[ansc](#)

**Examples**

```
##
varvalues<-afgen(xgrid=seq(0,1,length=21),ygrid=seq(0,1,length=21),samples=1000,binsize=32)

##creates 1000 samples of the two random variables zeta_B and A for each point
##(x,y) for x and y regularly-spaced probability vectors of length 21.
##
```

---

ansc

*Anscombe transformation*

---

**Description**

Does Anscombe's inverse sine transformation on a vector input.

**Usage**

```
ansc(x, binsize)
```

**Arguments**

x                   input data vector  
binsize            the binomial size corresponding to the observed binomial values.

**Details**

Performs the Anscombe calculation:  $A = \sin^{-1} \sqrt{(x + 3/8)/(binsize + 3/4)}$ .

**Value**

y                   vector of transformed data corresponding to x.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**References**

Anscombe, F.J. (1948) The transformation of poisson, binomial and negative binomial data. *Biometrika*, **35**, 246-254.

**See Also**

[afgen](#), [hfdnoise](#), [hfdnoise.wav](#), [link{invansc}](#)

**Examples**

```
#generate binomial data:
x<-rbinom(100,10,.5)
y<-ansc(x,10)
#this is now the transformed data.
```

---

asymean

*Asymptotic mean calculation*

---

**Description**

This function gives values for the asymptotic mean of the new binomial Fisz random variable for a grid of bivariate proportion values.

**Usage**

```
asymean(xgrid = seq(0, 1, length = 21), ygrid = seq(0, 1, length = 21), binsize = 32)
```

**Arguments**

xgrid            vector of x co-ordinate probabilities.  
ygrid            vector of y co-ordinate probabilities.  
binsize          the binomial size of the binomial random variables.

**Details**

See [afgen](#) for an explanation of the computation.

**Value**

zetam1m2        A matrix of dimension  $\text{length}(\text{xgrid}) \times \text{length}(\text{ygrid})$  of values of the mean.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**References**

Fisz, M. (1955), The Limiting Distribution of a Function of Two Independent Random Variables and its Statistical Application, *Colloquium Mathematicum*, **3**, 138–146.

**See Also**

[asyvar](#), [afgen](#)

**Examples**

```
means<-asymean(xgrid=seq(0,1,length=21),ygrid=seq(0,1,length=21),binsize=32)

## this produces a 21x21 matrix for an equally-spaced grid of binomial proportions.
```

---

asyvar

*Asymptotic variance function*

---

**Description**

This function gives values for the asymptotic mean of the new binomial Fisz random variable.

**Usage**

```
asyvar(xgrid = seq(0, 1, length = 21), ygrid = seq(0, 1, length = 21))
```

**Arguments**

xgrid            vector of x co-ordinate probabilities.  
ygrid            vector of y co-ordinate probabilities.

**Details**

Due to the form of the asymptotic variance for equal binomial sizes, this does not need a specification of the binomial size binsize (see [asymean](#)).

**Value**

asyvar            A matrix of dimension length(xgrid)xlength(ygrid) of values of the variance.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**References**

Fisz, M. (1955), The Limiting Distribution of a Function of Two Independent Random Variables and its Statistical Application, *Colloquium Mathematicum*, **3**, 138–146.

**See Also**

[asymean](#), [statgen](#)

**Examples**

```
variance<-asyvar(xgrid=seq(0,1,length=21),ygrid=seq(0,1,length=21))
## this produces a 21x21 matrix for an equally-spaced grid of binomial proportions.
```

---

binhf.wd

*Binomial Haar-Fisz wavelet transform*


---

**Description**

Forward Haar-Fisz transform for binomial random variables.

**Usage**

```
binhf.wd(x, binsize = 1, print.info=FALSE)
```

**Arguments**

x	data vector of binomial observations, of length a power of two.
binsize	the binomial size corresponding to x.
print.info	boolean to print some information about the coefficients.

**Details**

The procedure performs the Haar wavelet transform on the data x, and then modifies the wavelet coefficients by  $f_{jk} = d_{jk} / \sqrt{c_{jk} * (N - c_{jk}) / 2N}$ . The inverse Haar transform is then performed. This modification will stabilize the variance of the resulting vector.

**Value**

1	a list of two components transformed: transformed observations corresponding to x and cnew: scaling coefficient vector used in Fisz modification. This needs to be passed on to <code>invbinhf.wd</code> .
---	--

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

## References

Nunes, M.A. and Nason, G.P. (2009) A Multiscale Variance Stabilization for binomial sequence proportion estimation, *Statistica Sinica*, **19**(4), 1491-1510.

## See Also

[invbinhf.wd](#)

## Examples

```
x<-rbinom(256,32,.35)
```

```
y<-binhf.wd(x,32)
```

---

Blocks

*Proportion Functions*

---

## Description

An example Bernoulli proportion function.

## Usage

```
Blocks(x)
```

## Arguments

x a sequence of ‘time points’ as input into the function.

## Details

A proportion function based on the blocks function of Donoho, or that of Antoniadis and LeBlanc (2000). The extra “r” versions of these functions are reflected at the right endpoint.

## Value

y a vector of function values for the proportion function, corresponding to x.

## Author(s)

Matt Nunes (<m.nunes@ucl.ac.uk>)

## References

Antoniadis, A. and LeBlanc, F. (2000) Nonparametric wavelet regression for binary response. *Statistics*, **34**, 183–213.

**Examples**

```
t<-seq(0,1,length=256)
y<-Blocks(t)
plot(t,y, type="l")
```

---

chr20

*DNA datasets*

---

**Description**

Example DNA sequences.

**Usage**

```
data(chr20)
```

**Details**

The datasets are the chromosome 20 sequence of the human genome, and the mhc dataset available from the Human Genome Project website, binary-coded by base pair content and curtailed to a power of two.

**Source**

<http://www.sanger.ac.uk>

---

ebayesthresh.wavelet.wd

*Modified EbayesThresh wavelet thresholding function*

---

**Description**

Modified EbayesThresh functions.

**Details**

For help on these function, see the original help file supplied with the WaveThresh package. There is a modification to try and avoid zero noise standard deviation estimation.



---

free	<i>Freeman-Tukey transform</i>
------	--------------------------------

---

**Description**

Does Freeman-Tukey average inverse sine transformation on a vector input.

**Usage**

```
free(x, n)
```

**Arguments**

x	input data vector
n	the binomial size corresponding to the observed binomial values.

**Value**

a	vector of transformed data corresponding to x.
---	--

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**References**

Freeman, M. F. and Tukey, J. W. (1950) Transformations related to the angular and the square root. *Ann. Math. Stat.*, **21**, 607–611.

**See Also**

[freeinv](#)

**Examples**

```
#generate binomial data:
x<-rbinom(100,10,.5)
y<-free(x,10)
#this is now the transformed data.
```

freeinv

*Inverse Freeman-Tukey transform*

---

**Description**

Does the inverse of the Freeman-Tukey inverse sine transformation on a vector input.

**Usage**

```
freeinv(y, n)
```

**Arguments**

`y` input data vector.  
`n` the binomial size corresponding to the observed binomial values.

**Value**

`a` vector of transformed data corresponding to `y`.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**References**

Freeman, M. F. and Tukey, J. W. (1950) Transformations related to the angular and the square root. *Ann. Math. Stat.*, **21**, 607–611.

**See Also**

[free](#)

**Examples**

```
#generate binomial data:
x<-rbinom(100,10,.5)
y<-free(x,10)
x1<-freeinv(y,10)

#this should be the original data.
```

---

hf.inv2	<i>Haar-NN inverse transform</i>
---------	----------------------------------

---

**Description**

Inverse Haar-NN transform for binomial random variables ("in-place").

**Usage**

```
hf.inv2(data, binsize = 1)
```

**Arguments**

data	data vector of binomial observations, of length a power of two.
binsize	the binomial size corresponding to x.

**Details**

The procedure performs the inverse "in-place" Haar-NN wavelet transform on the data x.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**References**

Nunes, M.A. and Nason, G.P. (2009) A Multiscale Variance Stabilization for binomial sequence proportion estimation, *Statistica Sinica*, **19** (4), 1491–1510.

**See Also**

[invbinhf.wd](#)

---

hfdnoise	<i>Simulation function</i>
----------	----------------------------

---

**Description**

Proportion estimation procedure for simulations.

**Usage**

```
hfdnoise(n = 256, proportion = P2, binsize = 1, thrule = "ebayesthresh",  
         van = 8, fam = "DaubLeAsymm", pl = 3, prior = "laplace", vscale = "independent",  
         plotstep = FALSE, truncate = FALSE, ...)
```

**Arguments**

n	Length of vector to be sampled.
proportion	The function name of the proportion to be sampled.
binsize	The binomial size corresponding to the mean function proportion.
thrule	Thresholding procedure to be used in the smoothing. Possible values are "sureshrink" and "ebayesthresh".
van	the vanishing moments of the decomposing wavelet basis.
fam	the wavelet family to be used for the decomposing transform. Possible values are "DaubLeAsymm" and "DaubExPhase".
p1	the primary resolution to be used in the wavelet transform.
prior	Prior to be used in ebayesthresh thresholding.
vscale	argument to ebayesthresh thresholding procedure (variance calculation: "independent" or "bylevel").
plotstep	Should all steps be plotted in estimation procedure?
truncate	Should the estimates be truncated to lie in [0,1]?
...	Any other optional arguments.

**Details**

This function creates a regularly-spaced vector on the unit interval of length length, and uses these values to create corresponding values using the proportion function. These values are then used as binomial probabilities to sample "observed" binomial random variables. The observation vector is then denoised using a wavelet transform defined by the arguments p1, van, fam with thresholding method thrule. This denoising is done for both Anscombe and the Haar-Fisz method for binomial random variables. The procedure is repeated times times, and the resulting proportion estimates averaged.

**Value**

x	regular grid on which the proportion function is evaluated.
truep	vector corresponding to x of proportion function values.
fhat	Binomial Haar-Fisz estimate.
fhata	Anscombe inverse sine estimate.
fhatf	Freeman-Tukey average inverse sine estimate.
f11	lokern estimate using binhf.wd as a preprocessor.
f12	lokern estimate using Anscombe as a preprocessor.
bbwd	wd object of binomial Haar-Fisz before thresholding.
awd	wd object of Anscombe before thresholding.
b	data from which estimates were computed (sampled from truep).
bb	data after being preprocessed with binomial Haar-Fisz.
thr	Thresholded wd object of bbwd.
tmp	Thresholded (binomial Haar-Fisz) data before postprocessing.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**See Also**

[simsij](#)

**Examples**

```
sim<-hfdenoise()

plot(sim$x,sim$truep,type="l", xlab="",ylab="Binomial Proportion")

##^ shows original proportion to estimate.

lines(sim$x,sim$fhat,col=2)
lines(sim$x,sim$fhata,col=3)

##^ shows the estimates of the proportion from the two transforms.
```

---

hfdenoise.wav

*Denoising function*

---

**Description**

Denoise algorithm for thresholding methods supplied with wavethresh.

**Usage**

```
hfdenoise.wav(x, binsize, transform = "binhf", meth = "u", van = 1, fam = "DaubExPhase",
min.level = 3,coarse=FALSE)
```

**Arguments**

x	vector of observed values, of length a power of two.
binsize	the binomial size of the observed values x.
transform	A Gaussianizing transform. Possible values are "binhf" or "ansc".
meth	A wavelet thresholding method. Possible values are "u" for universal thresholding, or "c" for cross-validation.
van	the number of vanishing moments of the wavelet used in the wavelet denoiser.
fam	the wavelet family used in the wavelet denoiser. Possible values are "DaubLeAsymm" and "DaubExPhase".
min.level	the primary resolution level for the wavelet transform denoiser.
coarse	Boolean variable indicating whether a "coarsening" modification should be applied. For use with the chromosome datasets.

**Details**

The function pre and post-processes the observed data with either Anscombe's transform or the binomial Haar-Fisz transform, using a wavelet denoiser to smooth the data, specified by the inputs `min.level`, `van` and `fam` combined with the thresholding rule `meth`. If `coarse` is set to true, the first finest 11 coefficient levels are set to zero, corresponding to coefficients produced from  $2^{11} = 2048$  nucleotide bases.

**Value**

`fhat` vector corresponding to `x` of the estimated binomial proportion.

**Note**

This function requires the package `wavethresh`.

**Author(s)**

Matt Nunes (<[m.nunes@ucl.ac.uk](mailto:m.nunes@ucl.ac.uk)>)

**See Also**

[hfdenoise](#)

**Examples**

```
library(wavethresh)

#create a sample intensity vector:

int<-sinlog(seq(0,1,length=256))
x<-NULL
for(i in 1:256){
  x[i]<-rbinom(1,1,int[i])
}

est<-hfdenoise.wav(x,1,transform="ansc","u",6,"DaubLeAsymm",3,FALSE)
```

---

ht

*Forward Haar wavelet transform*

---

**Description**

Forward Haar transform.

**Usage**

`ht(x)`

**Arguments**

x data vector of (binomial) observations, of length a power of two.

**Details**

The procedure performs the Haar wavelet transform on the data x.

**See Also**

[ht.inv](#)

**Examples**

```
x<-rbinom(256,32,.35)
ht(x)
```

---

ht.inv

*Inverse Haar-NN*

---

**Description**

Inverse Haar transform for binomial random variables.

**Usage**

```
ht.inv(data)
```

**Arguments**

data transformed (binomial) observations: can be a list output from ht2 or a vector (finest details to coarsest, scaling coefficient).

**Details**

The procedure performs the inverse Haar wavelet transform.

**Value**

res datapoints in the function domain.  
sm1 smooth coefficients during the inverse transform.

**References**

Nunes, M.A. and Nason, G.P. (2009) A Multiscale Variance Stabilization for binomial sequence proportion estimation, *Statistica Sinica*, **19** (4), 1491–1510.

**See Also**

[ht2](#)

**Examples**

```
x<-rbinom(256,32,.35)
hx<-ht2(x)
y<-ht.inv(x)
```

---

invansc

*Inverse Anscombe transformation*

---

**Description**

Does the inverse of Anscombe's inverse sine transformation on a vector input.

**Usage**

```
invansc(y, n)
```

**Arguments**

y                   input data vector.  
n                   the binomial size corresponding to the observed binomial values.

**Value**

x                   vector of transformed data corresponding to y.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**References**

Anscombe, F.J. (1948) The transformation of poisson, binomial and negative binomial data. *Biometrika*, **35**, 246-254.

**See Also**

[ansc](#), [hfdenoise](#), [hfdenoise.wav](#)

**Examples**

```
#generate binomial data:
x<-rbinom(100,10,.5)
y<-ansc(x,10)
x1<-invansc(y,10)
#this should be the original data.
```



---

invbinhf.wd	<i>Inverse Haar-NN transform</i>
-------------	----------------------------------

---

**Description**

Performs the inverse Haar-NN transform for binomial random variables.

**Usage**

```
invbinhf.wd(transformed, binsize = 1, print.info=FALSE)
```

**Arguments**

transformed	a list of two components transformed: transformed observations of length a power of two and cnew: scaling coefficient vector used in Fisz modification.
binsize	the binomial size corresponding to the vector transformed.
print.info	boolean to print some information about the coefficients.

**Details**

The procedure performs the Haar wavelet transform on the data transformed, and then modifies the wavelet coefficients by  $d'_j k = d_j k * \sqrt{c_j k (N - c_j k) / 2N}$ . The inverse Haar transform is then performed. This modification will stabilize the variance of the resulting vector.

**Value**

estimate	a vector of transformed observations corresponding to transformed.
----------	--

**Note**

This function requires the package wavethresh.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**References**

Nunes, M.A. and Nason, G.P. (2009) "A Multiscale Variance Stabilization for binomial sequence proportion estimation", *Statistica Sinica*, **19** (4), 1491–1510.

**See Also**

[binhf.wd](#)

**Examples**

```
x<-rbinom(256,32,.35)
y<-binhf.wd(x,32)
x1<-invbinhf.wd(y,32)
```

---

norm

*Euclidean norm*

---

**Description**

Calculates the root squared error of two vectors.

**Usage**

```
norm(x,y)
```

**Arguments**

x	input data vector
y	input data vector

**Value**

e	error between the two input vectors
---	-------------------------------------

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**Examples**

```
#generate data:
x<-y<-runif(100)
error<-norm(x,y)

#this is the difference between the vectors.
```

---

pintens	<i>pintens</i>
---------	----------------

---

**Description**

An example binomial intensity vector.

**Usage**

```
data(pintens)
```

**Format**

The format is: num [1:1024] 0.278 0.278 0.278 0.278 0.278 ...

**Details**

The intensity is a vector of length 1024, based on a scaled ‘bumps’ function of Donoho and Johnstone.

**Examples**

```
data(pintens)
plot(pintens, type="l")
```

---

plotest	<i>Plotting function</i>
---------	--------------------------

---

**Description**

Plotting function for proportion estimates procedure.

**Usage**

```
plotest(l, plot.it = FALSE, verbose = FALSE)
```

**Arguments**

l	A results list from doall.
plot.it	Should results be plotted?
verbose	Should extra information be given during the procedure?

**Details**

This function uses norm to compute errors for estimates produced by doall.

**Value**

hfn	error between Haar-Fisz estimate and truep of doall.
an	error between Anscombe estimate and truep of doall.
fn	error between Freeman-Tukey estimate and truep of doall.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**See Also**

[norm](#)

**Examples**

```
sim<-hfdenoise()
plotest(sim)
```

---

propest.wav

*Proportion estimation function*

---

**Description**

Proportion estimation procedure for simulations.

**Usage**

```
propest.wav(proportion = P2, binsize=1,length = 256, times = 100, meth = "u", van = 6,
fam = "DaubLeAsymm", min.level = 3)
```

**Arguments**

proportion	A Bernoulli proportion/binomial mean function. Examples are P2, P4 and sinlog.
binsize	The binomial size corresponding to the mean function proportion.
length	Length of vector to be produced. Must be a power of two.
times	The number of times to sample the proportion.
meth	A wavelet thresholding method. Possible values are "u" for universal thresholding, or "c" for cross-validation.
van	the number of vanishing moments of the wavelet used in the wavelet denoiser.
fam	the wavelet family used in the wavelet denoiser. Possible values are "DaubLeAsymm" and "DaubExPhase".
min.level	the primary resolution level for the wavelet transform denoiser.

**Details**

This function creates a regularly-spaced vector on the unit interval of length `length`, and uses these values to create corresponding values using the proportion function. These values are then used as binomial probabilities to sample "observed" binomial random variables. The observation vector is then denoised using a wavelet transform defined by the arguments `van`, `fam`, `min.level` with thresholding method `meth`. This denoising is done for both Anscombe and the Haar-Fisz method for binomial random variables. The procedure is repeated `times` times, and the resulting proportion estimates averaged.

**Value**

<code>x</code>	regular grid on which the proportion function is evaluated.
<code>y</code>	vector corresponding to <code>x</code> of proportion function values.
<code>b</code>	matrix of dimensions <code>times</code> <code>x</code> <code>length</code> of sampled binomial variables.
<code>e</code>	matrix of dimensions <code>times</code> <code>x</code> <code>length</code> of estimated values of the proportion function, for the binomial Haar-Fisz transform.
<code>ea</code>	matrix of dimensions <code>times</code> <code>x</code> <code>length</code> of estimated values of the proportion function, for Anscombe's transform.
<code>meanfhat</code>	averaged proportion estimate for the binomial Haar-Fisz transform.
<code>meanfhata</code>	averaged proportion estimate for Anscombe's transform.
<code>amse</code>	average mean square error for the binomial Haar-Fisz transform.
<code>amsea</code>	average mean square error for Anscombe's transform.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**Examples**

```
## Not run:
sim<-propest.wav(proportion = P2, binsize=1,length = 256, times = 1000, meth = "u",
van = 6, fam = "DaubeAsymm", min.level = 4)

plot(sim$x,sim$y,type="l",xlab="",ylab="Binomial mean function")

##^ shows original proportion to estimate.

lines(sim$x,sim$meanfhat,col=2)
lines(sim$x,sim$meanfhata,col=3)

##^ shows the estimates of the proportion from the two transforms.

## End(Not run)
```

qqnormy

*Quantile generator*

---

**Description**

A Q-Q value generator.

**Usage**

```
qqnormy(y)
```

**Arguments**

y                    data sample

**Details**

This is an equivalent to qqnorm, but returning sorted values. See qqnorm.

**Value**

y                    vector of quantile values.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**See Also**

[qqstuff](#)

---

qqstuff*Quantile-quantile information about Haar-NN and Anscombe samples*

---

**Description**

A function to generate Q-Q plots (from simulations) for the Anscombe and (binomial) Haar-Fisz transforms.

**Usage**

```
qqstuff(intensity, binsize = 4, paths = 100, respaths = 1000, plot.q = FALSE,  
plot.sq = FALSE)
```

**Arguments**

<code>intensity</code>	an Bernoulli intensity vector, e.g. <code>pintens</code> .
<code>binsize</code>	a binomial size to generate a binomial mean vector.
<code>paths</code>	the number of paths sampled from the mean vector to use in Q-Q calculations.
<code>respaths</code>	the number of residual paths to use in squared residual calculations.
<code>plot.q</code>	A boolean variable, indicating whether simulation Q-Q plots should be outputted or not.
<code>plot.sq</code>	A boolean variable, indicating whether simulation squared residual plots should be outputted or not.

**Details**

`respaths` paths are sampled from the mean intensity vector. From these, the first `paths` are used to generate Q-Q data, which are then averaged for the Q-Q plots. The original paths are used to calculate a squared residual vector corresponding to the mean intensity vector.

**Value**

<code>qqinfo</code>	A 8 component list of quantile and residual plot information.
<code>vmat</code>	A matrix of dimensions <code>respathsxlength(intensity)</code> , each row being a path from the intensity vector.
<code>Av</code>	A matrix of dimensions <code>respathsxlength(intensity)</code> , each row an Anscombe-transformed path.
<code>bfv</code>	A matrix of dimensions <code>respathsxlength(intensity)</code> , each row a binomial Haar-Fisz-transformed path.
<code>vminus1</code>	A matrix of the difference between the paths and the mean intensity.
<code>vminus1</code>	A matrix of the difference between the Anscombe-transformed paths and the mean intensity.
<code>vminus1</code>	A matrix of the difference between the binomial Haar-Fisz-transformed paths and the mean intensity.
<code>Asqres</code>	vector of squared residuals of Anscombe-transformed paths.
<code>bfsqres</code>	vector of squared residuals of binomial Haar-Fisz-transformed paths.

**Note**

This function requires the package `wavethresh`. N.B. Since this function returns a lot of information, assign the output to a variable, to avoid printing endless information in the console.

**Author(s)**

Matt Nunes (<[m.nunes@ucl.ac.uk](mailto:m.nunes@ucl.ac.uk)>)

**See Also**

[qqnormy](#)

**Examples**

```
data(pintens)

a<-qqstuff(intensity=pintens,binsize=4,paths=100,respaths=100,plot.q=TRUE,plot.sq=TRUE)

#plots some interesting graphs.
```

---

shift                      *Shift function*

---

**Description**

This function shifts a vector input a certain number of places in the direction desired.

**Usage**

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

**Arguments**

v	a vector of input values.
places	the number of places to shift v.
dir	The direction to shift v.

**Details**

The function shifts the vector v by places in the direction of direction, using wrapping at the boundaries. Used for cycle spinning.

**Value**

vnew	the shifted version of v.
------	---------------------------

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**Examples**

```
v<-runif(10)

#have a look at v:

v

#now shift the values 4 places to the right...

shift(v,4,dir="right")
```



---

simsij                      *Simulation function*

---

### Description

Proportion estimation procedure for simulations.

### Usage

```
simsij(nsims = 100, n = 256, proportion = P2, binsize = 1,
       thrule = "ebayesthresh", van = 8, fam = "DaubLeAsymm", pl = 3,
       prior = "laplace",
       vscale = "independent", plotstep = FALSE, a = NA, truncate = FALSE, ...)
```

### Arguments

nsims	The number of times to repeat the function doall (on random datasets from proportion).
n	Length of vector to be sampled.
proportion	The function name of the proportion to be sampled.
binsize	The binomial size corresponding to the mean function proportion.
thrule	Thresholding procedure to be used in the smoothing. Possible values are "sureshrink" and "ebayesthresh".
van	the vanishing moments of the decomposing wavelet basis.
fam	the wavelet family to be used for the decomposing transform. Possible values are "DaubLeAsymm" and "DaubExPhase".
pl	the primary resolution to be used in the wavelet transform.
prior	Prior to be used in ebayesthresh thresholding.
vscale	argument to ebayesthresh thresholding procedure (variance calculation: "independent" or "bylevel").
plotstep	Should all steps be plotted in estimation procedure?
a	the a argument for EbayesThresh.
truncate	Should the estimates be truncated to lie in [0,1]?
...	Any other optional arguments.

### Details

This function creates a regularly-spaced vector on the unit interval of length length, and uses these values to create corresponding values using the proportion function. These values are then used as binomial probabilities to sample "observed" binomial random variables. The observation vector is then denoised using a wavelet transform defined by the arguments van, fam, min.level with thresholding method meth. This denoising is done for both Anscombe and the Haar-Fisz method for binomial random variables. The procedure is repeated times times, and the resulting proportion estimates averaged.

**Value**

x	regular grid on which the proportion function is evaluated.
truep	vector corresponding to x of proportion function values.
ans	matrix containing the errors from each of the nsims doall runs.
est	Array containing the nsims estimates produced by Anscombe and Haar-Fisz.
bin	Matrix of the raw binomial samples for each of the nsims runs.

**Author(s)**

Matt Nunes (<m.nunes@ucl.ac.uk>)

**See Also**

[hfdenoise](#)

**Examples**

```
## Not run:
a<-simsij(nsims=100)

plot(a$est[1,,1])

##^^ shows 1st binomial Haar-Fisz estimate.
## End(Not run)
```

---

statgen

*Statistics generator*

---

**Description**

This function generates useful simulation statistics for NN and Anscombe transforms.

**Usage**

```
statgen(valuelist, xgrid = seq(0, 1, length = 21), ygrid = seq(0, 1, length = 21),
binsize = 32, plot.m = FALSE, plot.v = FALSE, plot.ks = FALSE, ptype = "persp")
```

**Arguments**

valuelist	a two component list as produced by afgen.
xgrid	a vector of x coordinate binomial proportions.
ygrid	a vector of x coordinate binomial proportions.
binsize	binomial size to use in simulations.

plot.m	A boolean variable, indicating whether mean simulation plots should be outputted.
plot.v	A boolean variable, indicating whether variance simulation plots should be outputted.
plot.ks	A boolean variable, indicating whether Kolmogorov-Smirnov simulation plots should be outputted.
ptype	where appropriate, the type of plots to be produced. Possible values are "persp" for 3D perspective plots or "contour" for corresponding contour plots.

### Details

The function does several sample variance plots, Kolmogorov-Smirnov and mean plots for the data in the variable `valuelist` (for both Anscombe and binomial Haar-Fisz transforms).

### Value

afm	matrix of sample mean values for binomial Haar-Fisz samples.
anm	matrix of sample mean values for Anscombe samples.
afv	matrix of sample variance values for binomial Haar-Fisz samples.
anv	matrix of sample variance values for Anscombe samples.
afk	matrix of Kolmogorov-Smirnov statistics for binomial Haar-Fisz samples.
ank	matrix of Kolmogorov-Smirnov statistics for Anscombe samples.

### Author(s)

Matt Nunes (<m.nunes@ucl.ac.uk>)

### See Also

[afgen](#)

### Examples

```
a<-afgen(xgrid = seq(0, 1, length = 21), ygrid = seq(0, 1, length = 21),
samples = 1000, binsize = 32)
```

```
b<-statgen(a,xgrid=seq(0,1,length=21),ygrid=seq(0,1,length=21),binsize=32,plot.m=FALSE,
plot.v=TRUE,plot.ks=FALSE,ptype="persp")
```

# Index

- \* **datagen**
  - afgen, 2
  - Blocks, 7
  - qqnormy, 22
  - qqstuff, 22
- \* **datasets**
  - chr20, 8
  - pintens, 19
- \* **manip**
  - ansc, 3
  - asymean, 4
  - asyvar, 5
  - binhf.wd, 6
  - free, 9
  - freeinv, 10
  - hf.inv2, 11
  - ht, 14
  - ht.inv, 15
  - invansc, 16
  - invbinhf.wd, 17
  - norm, 18
  - shift, 24
  - statgen, 26
- \* **regression**
  - ebayesthresh.wavelet.wd, 8
  - hfdenoise, 11
  - hfdenoise.wav, 13
  - plotest, 19
  - propest.wav, 20
  - simsij, 25
- afgen, 2, 4, 5, 27
- ansc, 3, 3, 16
- asymean, 4, 5, 6
- asyvar, 5, 5
- binhf.wd, 6, 17
- Blocks, 7
- Blocksr (Blocks), 7
- Bumps (Blocks), 7
- Bumpsr (Blocks), 7
- Bursts (Blocks), 7
- Burstsr (Blocks), 7
- chr20, 8
- const (Blocks), 7
- ebayesthresh.wavelet.wd, 8
- free, 9, 10
- freeinv, 9, 10
- hf.inv2, 11
- hfdenoise, 4, 11, 14, 16, 26
- hfdenoise.wav, 4, 13, 16
- ht, 14
- ht.inv, 15, 15
- ht2, 15
- ht2 (ht), 14
- invansc, 16
- invbinhf.wd, 7, 11, 17
- madmad (ebayesthresh.wavelet.wd), 8
- mhc (chr20), 8
- negloglik.laplace  
(ebayesthresh.wavelet.wd), 8
- norm, 18, 20
- P2 (Blocks), 7
- P3 (Blocks), 7
- P3a (Blocks), 7
- P4 (Blocks), 7
- pintens, 19
- plotest, 19
- propest.wav, 20
- qqnormy, 22, 23
- qqstuff, 22, 22
- shift, 24

simsij, [13](#), [25](#)

sinlog (Blocks), [7](#)

sinlogr (Blocks), [7](#)

statgen, [6](#), [26](#)

threshold.wd (ebayesthresh.wavelet.wd),  
[8](#)

wandafromx (ebayesthresh.wavelet.wd), [8](#)