Package ‘hwwntest’

October 13, 2022

Type Package

Title Tests of White Noise using Wavelets

Version 1.3.1

Date 2018-08-03

Description Provides methods to test whether time series is consistent
with white noise.

Depends R(>= 3.0)

Imports parallel, polynom, wavethresh

License GPL-2

NeedsCompilation no

Author Delyan Savchev [aut],
Guy Nason [aut, cre]

Maintainer Guy Nason <g.p.nason@bristol.ac.uk>

Repository CRAN

Date/Publication 2018-08-03 11:10:07 UTC

R topics documented:

`hwwntest-package` *Tests of White Noise using Wavelets*

Description

Provides methods to test whether time series is consistent with white noise.

Details

The DESCRIPTION file:

Package:	hwwntest
Type:	Package
Title:	Tests of White Noise using Wavelets
Version:	1.3.1
Date:	2018-08-03
Authors@R:	c(person("Delyan", "Savchev", role=c("aut"),email="madbss@bristol.ac.uk"), person("Guy", "Nason", role=c(
Description:	Provides methods to test whether time series is consistent with white noise.
Depends:	R(>= 3.0)
Imports:	parallel, polynom, wavethresh
License:	GPL-2
Author:	Delyan Savchev [aut], Guy Nason [aut, cre]
Maintainer:	Guy Nason <g.p.nason@bristol.ac.uk>

Index of help topics:

```
Macdonald               Compute the Macdonald density function for a
                        specified parameter value 'm' at a vector of
                        'x' values.
bartlettB.test          Bartlett's B test for white noise
compute.rejection       Function to compute empirical size or power for
                        various tests of white noise.
cumperiod               Compute cumulative normalized periodogram.
d00.test                Test for white noise based on the coarsest
                        scale Haar wavelet coefficient of the spectrum.
genwwn.powerplot        Plot (approximation) to the theoretical power
                        of the 'genwwn.test' test for ARMA processes
                        (including, of course, white noise itself) for
                        a range of sample sizes.
genwwn.test             White noise test using general wavelets.
genwwn.thpower          Compute (approximation) to the theoretical
```

	power of the 'genwwn.test' test for ARMA processes (including, of course, white noise itself).
hwwn.dw	Compute discrete wavelets
hwwn.test	Perform a test for white noise on a time series.
hwwntest-package	Tests of White Noise using Wavelets
hywavwn.test	Hybrid wavelet test of white noise.
hywn.test	Hybrid of Box-Ljung test, Bartlett B test, Haar wavelet and General wavelet tests.
sqcoefvec	Compute coefficients required for approximaing the wavelet transform using the square of wavelets.
sqndwd	Compute the non-decimated squared wavelet transform.
sqndwdecomp	Brute-force calculation of the non-decimated squared wavelet transform.
sqwd	Compute expansion with respect to squared wavelets.

Contains a variety of hypothesis tests for white noise data. The package contains an implementation of Bartlett's B test, `bartlettB.test`, (Kolmogorov-Smirnov test on the cumulative periodogram), a selection of wavelet-based tests `hwwn.test` a test using Haar wavelets, `d00.test` a single Haar wavelet coefficient test, `genwwn.test` a test using smoother Daubechies wavelets, a hybrid test `hywavwn.test` that uses Haar wavelets at fine scales and general wavelets at coarse scales and a omnibus test `hywn.test` that combines the results of four tests (`hwwn.test`, `genwwn.test`, `bartlettB.test` and the `Box.test`) The wavelet tests work by examining the wavelet transform of the regular periodogram and assess whether it has non-zero coefficients. If series is H_0: white noise, then the underlying spectrum is constant (flat) and all true wavelet coefficients will be zero. Then all periodogram wavelet coefficients will have true zero mean which can be tested using knowledge of, or approximation to, the coefficient distribution.

Author(s)

NA

Maintainer: NA

References

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. http://dx.doi.org/10.1002/sta4.69

See Also

`hwwn.test`

Examples

```
# Invent test data set which IS white noise
```

```
#
x <- rnorm(128)
#
# Do the test
#
x.wntest <- hwwn.test(x)
#
# Print the results
#
#x.wntest
#
#       Wavelet Test of White Noise
#
#data:
#p-value = 0.9606
#
# So p-value indicates that there is no evidence for rejection of
# H_0: white noise.
#
# Let's do an example using data that is not white noise. E.g. AR(1)
#
x.ar <- arima.sim(n=128, model=list(ar=0.8))
#
# Do the test
#
x.ar.wntest <- hwwn.test(x.ar)
#
# Print the results
#
print(x.ar.wntest)
#
#       Wavelet Test of White Noise
#
#data:
#p-value < 2.2e-16
#
# p-value is very small. Extremely strong evidence
# to reject H_0: white noise
#
#
# Let's use one of the other tests: e.g. the general wavelet one
#
x.ar.genwwntest <- genwwn.test(x.ar)
#
# Print the results
#
print(x.ar.genwwntest)
#
#
#  Wavelet Test of White Noise
#
# data:
# p-value = 1.181e-10
```

```
#
# Again, p-value is very small
```

`bartlettB.test` *Bartlett's B test for white noise*

Description

Bartlett's test uses the Kolmogorov-Smirnov test applied to the cumulative normalized periodogram.

Usage

```
bartlettB.test(x, plot.it = FALSE)
```

Arguments

`x`	The time series you wish to test, of any length.
`plot.it`	If TRUE then the normalized cumulative periodogram is plotted along with a straight line that indicates the theoretical line of this object under the null hypothesis. A further plot of the density of the true statistic under the null hypothesis is produced.

Details

This test: (i) computes the periodogram, (ii) derives the normalized cumulative periodogram using the `cumperiod` function. Under the null hypothesis of white noise the periodogram is a set of iid exponential random variables, asymptotically. So, the cumulative periodogram should look like a straight line at a 45 degree angle. The test statistic is the maximum deviation of the normalized cumulative periodogram and this straight line. The p-value of the test is computed within the function by the `b.power` function. This is an example of a Kolmogorov-Smirnov statistical test.

Value

An object of class `htest`. A list containing the following components:

`statistic`	The value of the Bartlett test statistic.
`p.value`	The p-value of the test
`method`	A text string saying what the method was

Note

Code was based on Professor Newton's explanation

Author(s)

G. P. Nason

References

Bartlett, M.S. (1967) Some Remarks on the Analysis of Time-Series. *J. R. Statist. Soc. B*, **54**, 25-38.

http://www.stat.tamu.edu/~jnewton/stat626/topics/topics/topic13.pdf Link to Professor H. Joseph Newton's web page on Bartlett's test

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. http://dx.doi.org/10.1002/sta4.69

See Also

compute.rejection, cumperiod

Examples

```
#
# Do white noise test on smallish data set
#
x <- rnorm(30)
bartlettB.test(x)
#
# For my realization the answer was:
#
#
# Bartlett B Test for white noise
#
#data:
#= 0.3747, p-value = 0.999
#
# So, we accept H_0
```

`compute.rejection` *Function to compute empirical size or power for various tests of white noise.*

Description

Can generate white noise sequences, or ARMA time series and subject multiple realizations of these to various tests for white noise. The function then counts how many have been rejected to give some idea of empirical size (if no `ar` or `ma` term is specified) or power (if such terms are specified).

Usage

```
compute.rejection(ar = NULL, ma = NULL, npow = 100, nom.size = 0.05,
ndata = 1024, lapplyfn = lapply, Box.lag = 1, rand.gen = rnorm,
hwwn = TRUE, box = TRUE, bartlett = TRUE,
d00test = TRUE, genwwn = TRUE, hywn = TRUE, hywavwn = TRUE,
filter.number = 10, family = "DaubExPhase",
away.from = "standard", ...)
```

Arguments

`ar`	Any autoregressive terms to go directly to the `arima.sim` function. Leave as it is if you wish to simulate white noise.
`ma`	As `ar` but for moving average terms.
`npow`	The number of realizations to carry out. The best assessments are carried out with high values of `npow`, e.g. 1000 or even 10000.
`nom.size`	The nominal statistical size of the test. Note: this does not change the nomimal size for ALL tests. You need to check each help pages for each function to check what can be changed.
`ndata`	The length of the white noise or ARMA realizations. Power for both these tests depends on sample size.
`lapplyfn`	If you have the library `parallel` and a suitable multicore machine then this function can run the realizations in parallel. If so, then you can change this argument to `lapplyfn=mclapply` to take advantage of this.
`Box.lag`	The Box test tests for white noise by examining autocorrelation coefficients. This argument specifies the max number of autocorrelation coefficients, ie. coefficients from lag 1 up to `Box.lag`.
`rand.gen`	Alternative innovation generator. By default Gaussian innovations are used, but you can specify alternatives to get heavy-tailed innovations, for example.
`hwwn`	If TRUE then the `hwwn.test` will be evaluated, if FALSE then it won't be.
`box`	If TRUE then the `Box.test` will be evaluated, if FALSE then it won't be.
`bartlett`	If TRUE then the `bartlettB.test` will be evaluated, if FALSE then it won't be.
`d00test`	If TRUE then the `d00.test` will be evaluated, if FALSE then it won't be.
`genwwn`	If TRUE then the `genwwn.test` will be evaluated, if FALSE then it won't be.
`hywn`	If TRUE then the `hywn.test` will be evaluated, if FALSE then it won't be.
`hywavwn`	If TRUE then the `hywavwn.test` will be evaluated, if FALSE then it won't be.
`filter.number`	The number of vanishing moments of wavelets used in the general wavelet tests (genwwn, hywn and hywavwn).
`family`	Wavelet family, as for `filter.number` argument.
`away.from`	The number of finer scales not to use for the general wavelet tests. These tests work by relying on the asymptotic normality of wavelet coefficients, but this only becomes useful away from the finer scales. This argument can be an integer in which case it defines the number of fine scales to ignore. Alternatively, you can supply the argument `"standard"` which chooses an automatically selected number of scales to stay away from which works well up to time series in length of 1000. Better performance can be obtained for series longer than 1000 by adapting the `away.from` argument.
`...`	Other arguments to `hwwn.test`.

Details

This function repeatedly runs the hypothesis tests on realizations from a stochastic process which can be white noise (if `ar` and `ma` are `NULL`) or an ARMA process specified by `ar` and `ma`. It then counts how many times the null was rejected and returns this as proportion of the total number of realizations. In this way, this function can compute the empirical size and power of the tests.

Value

A list with eight components. Each component is a number, between zero and one, which corresponds to the empirical size or power of each test. Note, if any component is `NULL` this means that it was not evaluated and was 'turned off' in the command line by setting its name equal to `FALSE`.

Author(s)

Delyan Savchev and Guy Nason

References

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. http://dx.doi.org/10.1002/sta4.69

See Also

`bartlettB.test`, `d00.test`, `genwwn.test`, `hwwn.test`, `hywn.test`, `hywavwn.test`

Examples

```
#
# Compute empirical size of both tests using 1000 realizations
# with data of length 32
#
answer <- compute.rejection(npow=100, ndata=32)
#
# Print the answer
#
print(answer)
#$hwwntest.rejprop
#[1] 0.03
#
#$box.rejprop
#[1] 0.02
#
#$bartlett.rejprop
#[1] 0.01
#
#$d00.pow
#[1] 0.03
#
#$genwwn.pow
#[1] 0.02
#
```

```
#$hywn.pow
#[1] 0.01
#
#$hywavwn.pow
#[1] 0.03
#
#
# So, all empirical sizes should be close to their nominal value of 0.05
#
# Now let's try and ascertain the empirical power on an AR(1)
#
answer <- compute.rejection(ar=0.8, npow=100, ndata=32)
#
# Print the answer
#
print(answer)
#$hwwntest.rejprop
#[1] 0.79
#
#$box.rejprop
#[1] 0.98
#
#$bartlett.rejprop
#[1] 0.97
#
#$d00.pow
#[1] 0.97
#
#$genwwn.pow
#[1] 0.94
#
#$hywn.pow
#[1] 0.95
#
#$hywavwn.pow
#[1] 0.85
#
# Most powers are pretty good.
```

cumperiod *Compute cumulative normalized periodogram.*

Description

Computes cumulative normalized periodogram.

Usage

```
cumperiod(x)
```

Arguments

x	The time series you wish to compute the cumulative normalized periodogram.

Details

Does as the title suggests

Value

A list containing the following two components:

wp	The Fourier frequencies where the cumulative normalized periodogram is evaluated at
cumperiod	The cumulative normalized periodogram.

Note

Code was based on Professor Newton's explanation

Author(s)

G. P. Nason

References

Bartlett, M.S. (1967) Some Remarks on the Analysis of Time-Series. *J. R. Statist. Soc. B*, **54**, 25-38.

Link to Professor H. Joseph Newton's web page on Bartlett's test

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. http://dx.doi.org/10.1002/sta4.69

See Also

bartlettB.test, compute.rejection

Examples

```
#
# Use example time series
#
x <- rnorm(100)

x.cp <- cumperiod(x)
#
# Can plot it, if you like
#
## Not run: plot(x.cp$wp, x.cp$cumperiod, type="l", xlab="Frequency",
ylab="Cumulative Normalized Periodogram")
## End(Not run)
#
```

```
# You can try replacing the x by, say, an AR(1) using arima.sim and
# you'll get a very different shaped line, depending on the AR(1)
# parameter.
```

`d00.test` *Test for white noise based on the coarsest scale Haar wavelet coefficient of the spectrum.*

Description

Computes the coarsest scale Haar wavelet coefficient of the periodogram but directly using a formula based on a particular linear combination of autocorrelation coefficients. Then performs a hypothesis test by comparing the test statistic to a standard normal distribution.

Usage

```
d00.test(x)
```

Arguments

`x`	The time series you want to test, of arbitrary length.

Value

An object of class `htest` containing the following components.

`statistic`	The test statistic
`p.value`	The p-value of the test
`method`	A test string indicating the method

Author(s)

Delyan Savchev and Guy Nason

References

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. `http://dx.doi.org/10.1002/sta4.69`

See Also

`compute.rejection`,`hwwn.test`

Examples

```
#
# Test data set
#
x <- rnorm(30)
#
#
answer <- d00.test(x)
#
# My answer was:
#
# d00 test on acfs
#
#data:
#= -1.696, p-value = 0.08989
```

`genwwn.powerplot` *Plot (approximation) to the theoretical power of the* `genwwn.test` *test for ARMA processes (including, of course, white noise itself) for a range of sample sizes.*

Description

Computes and plots (approximation) to the theoretical power of the `genwwn.test` test using the `genwwn.thpower` function.

Usage

```
genwwn.powerplot(N =c(32, 64, 128, 256, 512, 1024), ar = NULL,
ma = NULL, plot.it = TRUE, sigsq = 1, alpha = 0.05,
away.from = "standard", filter.number = 10,
family = "DaubExPhase", verbose = FALSE, ylim=c(0,1))
```

Arguments

`N`	Vector of lengths of the series you want to plot theoretical power for.
`ar`	Autoregressive parameters. A vector with p entries for AR(p) with the first entry being the value for lag-one term (alpha_1), the second entry being the value for the lag-two term (alpha_2) etc. If this argument is NULL then there are no AR terms.
`ma`	Similar to the `ar` argument except for MA terms. A vector of length q for MA(q) parameters, with first entry being beta_1, the second being beta_2, etc. If this argument is NULL then there are no MA terms.
`plot.it`	If `TRUE` then a plot of theoretical power against sample size is produced. The computed theoretical powers for the fixed sample sizes specified by `N` are plotted as crosses. The crosses are then joined by a dashed line to indicate a likely trajectory of the theoretical power for sample sizes not computed.

`sigsq`	The theoretical innovation variance (also the variance of white noise if `ar=ma=NULL`.
`alpha`	The nominal size of the test for this theoretical power calculation.
`away.from`	Describes how many fine scales to exclude, the same as in `genwwn.test`. This can be an integer up to the number of scales. However, mostly you can leave this at "standard" where the scales calculation is automatically determined.
`filter.number`	The number of vanishing moments in the Daubechies series of wavelets.
`family`	The wavelet family.
`verbose`	If TRUE then informative messages are printed during the progress of the function.
`ylim`	The theoretical power is a probability and lies in the range of zero to one and this argument specifies those limits for the vertical axis. These can be changed to whatever you like. E.g. if all the powers were similar (e.g. if the null series was white noise and `alpha`=0.05 then the theoretical powers would all be approximately 0.05.)

Details

Function calculates the value of the power function at the specified sample sizes using the `genwwn.thpower` function. Then these values are plotted and returned.

Value

A list containing the following components.

`N`	The vector of sample sizes.
`power`	The computed theoretical powers for each sample size
`ar`	The autoregressive parameters, NULL if there are none.
`ma`	The moving average parameters, NULL if there are none.
`sigsq`	The innovation variance used
`alpha`	The significance level used
`away.from`	The value of the `away.from` argument supplied
`filter.number`	The wavelet filter number used
`family`	The wavelet family used

Author(s)

Delyan Savchev and Guy Nason

References

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. `http://dx.doi.org/10.1002/sta4.69`

See Also

`genwwn.test`, `genwwn.thpower`

Examples

```
#
# Plot theoretical power for white noise
#
## Not run: genwwn.powerplot()
#
# Plot theoretical power for AR(1) process
#
## Not run: genwwn.powerplot(ar=0.8)
```

`genwwn.test` *White noise test using general wavelets.*

Description

Performs test for white noise using a general wavelet decomposition of a normalized periodogram.

Usage

```
genwwn.test(x, away.from = "standard", lowlev = 0, plot.it = FALSE,
stopeveryscale = FALSE, filter.number = 10,
family = "DaubExPhase", mc.method = p.adjust.methods,
mac.spread = 10, verbose = FALSE)
```

Arguments

`x`	The time series you wish to test (of dyadic length).
`away.from`	Number of fine scales to stay away from, see details below. If "standard" then this is automatically computed for sample sizes up to length of 1024. If you have a longer series then the test will still work but might not be quite as powerful (but probably not too bad either).
`lowlev`	The coarsest coefficient to evaluate. This should always be left at 0.
`plot.it`	If TRUE then a series of plots similar to the ones produced in the `hwwn.test` function is produced. See that help page for further details on what the plots show.
`stopeveryscale`	If TRUE then if `plot.it=TRUE` then a 'scan' is issued after every plot. Just hit RETURN to continue.
`filter.number`	The number of vanishing moments of the wavelet used to compute coefficients that are then evaluated to see whether they are zero. In principle, best compression for a sparse evaluation of the normalized spectrum should mean we use the smoothest wavelets with the highest number of vanishing moments which is ten. The other components of the function are optimized for ten vanishing moments. The function will still work for other numbers of vanishing moments but maybe with slightly reduced power.
`family`	Wavelet family to go with `filter.number`.

`mc.method`	The type of multiple hypothesis correction, see `p.adjust` for details.
`mac.spread`	Horizontal range for plotting of wavelet coefficients, only used if `plot.it=TRUE`.
`verbose`	If `TRUE` some information messages are printed.

Details

This function computes the normalized periodogram, and then subjects it to a wavelet transform with respect to any wavelet (in wavethresh). Under the null hypothesis of white noise the coefficients should all close to zero and this function works out, for each coefficient, how close statistically it is to zero by assuming a Gaussian null distribution with mean zero and variance one. Then the multiple p-values from each of these tests are adjusted for multiple hypothesis test by using the `p.adjust` function before returning an overall p-value for the test. The test has been optimized for using the `filter.number=10` wavelet and `away.from="standard"`, but should work pretty well for other wavelets and even away.from values of more than 2-3 for moderate numbers of scales, and potentially higher for longer data sets.

An approximation to the theoretical power of this test can be obtained using the `genwwn.thpower` function.

Value

An object of class `htest` with the following components.

`p.val.collector`	All the of unadjusted p-values
`p.val.adjust`	All of the adjusted p-values
`p.value`	The overall p-value of the test
`method`	A text string describing the test

Author(s)

Delyan Savchev and Guy Nason

References

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. http://dx.doi.org/10.1002/sta4.69

See Also

`compute.rejection`, `genwwn.thpower`

Examples

```
#
# Generate test set, of dyadic length
#
x <- rnorm(64)
#
# Do the test:
```

```
#
answer <- genwwn.test(x)
#
# What do we get?
#
#answer
#
# Wavelet Test of White Noise
#
#data:
#p-value = 0.4629
```

`genwwn.thpower` *Compute (approximation) to the theoretical power of the* `genwwn.test` *test for ARMA processes (including, of course, white noise itself).*

Description

Compute (approximation) to the theoretical power of the `genwwn.test` test. Note: this function does no simulation, it merely computes an approximation to the likely statistical power (or size) of the `genwwn.test` function. It can be useful to establish the reverse question: what sample size do I require to achieve a certain power for a given ARMA process?

Usage

```
genwwn.thpower(N = 128, ar = NULL, ma = NULL, plot.it = FALSE,
sigsq = 1, alpha = 0.05, away.from = "standard",
filter.number = 10, family = "DaubExPhase", verbose = FALSE)
```

Arguments

`N`	The length of the series you want to get a theoretical power result for.
`ar`	Autoregressive parameters. A vector with p entries for AR(p) with the first entry being the value for lag-one term (alpha_1), the second entry being the value for the lag-two term (alpha_2) etc. If this argument is NULL then there are no AR terms.
`ma`	Similar to the `ar` argument except for MA terms. A vector of length q for MA(q) parameters, with first entry being beta_1, the second being beta_2, etc. If this argument is NULL then there are no MA terms.
`plot.it`	If `TRUE` then two plots are produced. The first is of the time series spectrum you are considering (controlled by the `N`, `ar` and `ma` arguments.) The second is a plot of the wavelet coefficients of the normalized spectrum.
`sigsq`	The theoretical innovation variance (also the variance of white noise if `ar=ma=NULL`.
`alpha`	The nominal size of the test for this theoretical power calculation.

`away.from`	Describes how many fine scales to exclude, the same as in `genwwn.test`. This can be an integer up to the number of scales. However, mostly you can leave this at "standard" where the scales calculation is automatically determined.
`filter.number`	The number of vanishing moments in the Daubechies series of wavelets.
`family`	The wavelet family.
`verbose`	If TRUE then informative messages are printed during the progress of the function.

Details

Function calculates the value of the power function at the specified arguments. It does this by: (i) specifying the functional spectrum of the ARMA process (which can be flat, ie white noise); (ii) calculating the variance of the ARMA process by numerical integration of the spectrum; (iii) calculating the spectrum values at the Fourier frequencies; (iv) calculating the wavelet coefficients at the exact spectrum values; (v) computing the exact variance of the wavelet coefficients of the squared normalized spectrum; (vi) computing the approximate power of the whole lot.

Value

A list containing the following components.

`C.alpha.c`	The critical value for the test, which is the nominal size critical value after correction for multiple hypothesis tests (correction using Bonferroni).
`th.power`	The computed theoretical power
`norspecwd`	The wavelet coefficients of the true spectrum
`norspecvarwd`	The squared wavelet transform of the squared normalized spectrum
`all.hwc`	All of the wavelet coefficients from the normalized true specturm as a single vector
`all.sdwc`	The 'true' standard deviations of the wavelet coefficients

Author(s)

Delyan Savchev and Guy Nason

References

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. `http://dx.doi.org/10.1002/sta4.69`

See Also

`genwwn.test`, `sqwd`

Examples

```
#
# Calculate what the theoretical actual size is likely to be for the
# genwwn.test for a white noise sequence of T=64, nominal size=0.05
#
genwwn.thpower(N=64)$th.power
#[1] 0.04894124
#
# This is pretty close to the nominal size of 5%. Good.
#
# What is the power of detection for the AR(1) process with alpha=0.3?
# Let's say with sample size of T=32
#
genwwn.thpower(N=32, ar=0.3)$th.power
#[1] 0.2294128
#
# That's pretty poor, we'll only detect about 23% of cases. Can we achieve
# a power of 90%? Actually, it turns out that by repeating these above
# functions with N=128 gives a power of 61%, and for N=256 we get a power of
# 90%.
```

`hwwn.dw` *Compute discrete wavelets*

Description

Compute discrete wavelets up to some scale

Usage

```
hwwn.dw(J, filter.number, filter.family)
```

Arguments

`J`	The number of scales of discrete wavelets to produce.
`filter.number`	The wavelet filter number.
`filter.family`	The wavelet family to produce.

Details

Uses the delta value method and uses wr to reconstruct the wavelets. See Nason, von Sachs and Kroisandt 2000 for details.

Note: this function is the same as the `discrete.wavelets` function in the AutoSpec package, but copied here.

Value

A list of length J, each component of the list corresponds to a different scale of wavelets. Component 1 is the finest scale, component 2 is the next finest and so on.

Author(s)

Piotr Fryzlewicz

References

Nason, G.P., von Sachs, R. and Kroisandt, G. (2000) Wavelet processes and adaptive estimation of the evolutionary wavelet spectrum. *J. R. Statist. Soc. B*, **62**, 271-292.

See Also

`sqndwdecomp`

Examples

```
#
# Generate three scales of Haar wavelets
#
hwwn.dw(3, 1, "DaubExPhase")
#[[1]]
#[1] 0.7071068 -0.7071068
#
#[[2]]
#[1] 0.5 0.5 -0.5 -0.5
#
#[[3]]
#[1] 0.3535534 0.3535534 0.3535534 0.3535534 -0.3535534 -0.3535534 -0.3535534
#[8] -0.3535534
```

`hwwn.test` *Perform a test for white noise on a time series.*

Description

Often one wishes to know whether a time series is consistent with a white noise model. This function tests whether the underlying spectrum of the time series is flat, which is identical to saying that all the autocorrelations of the series are zero (apart from the lag zero autocorrelation which is always one). This test is exact for Gaussian data but will also work well with heavy-tailed distributions whose periodogram tends to the exponential distribution asymptotically (see accompanying paper for details).

Usage

```
hwwn.test(x, lowlev = 0, plot.it = FALSE, stopeveryscale = FALSE,
n.cdf.grid = 1000, mc.method = p.adjust.methods, mac.spread=10)
```

Arguments

`x`	The data set you wish to test. For now, the length of this series has to be a power of two. In theory, it could be any length.
`lowlev`	Specifies the coarsest resolution level of wavelet coefficients computed on the spectrum. Typically, this should be left at one, which is the coarsest that can be achieved an still approximate the CDF
`plot.it`	If `TRUE` then plots of the wavelet coefficients and their modelled underlying distribution are plotted, and their cumulative distributions and the resultant p-values as a histogram for each scale. Theoretical values are in red and data estimated values in black.
`stopeveryscale`	If `TRUE` the code stops after every plot if `plot.it==TRUE`. This is a way of ensuring that the human can see every plot to stop it whizzing off the screen. Simply press ENTER to continue.
`n.cdf.grid`	The CDF of the Macdonald distribution is evaluated numerically. This argument controls the resolution of that grid: it controls the number of grid points there are between `-mac.spread` and `mac.spread`.
`mc.method`	The method of multiple hypothesis comparison. See `p.adjust` for details.
`mac.spread`	The range (from `-mac.spread` to `mac.spread`) that the CDF of the Macdonald distribution is computed on.

Details

The null hypothesis of the test contained in this function is H_0: series is white noise (or constant spectrum) versus H_A: it is not white noise. This test works by assessing whether the spectrum of the underlying series is constant or not. It does this by first computing the periodogram of the sample series. This is a well-studied estimate of the spectrum. Then it evaluates the constancy of the spectrum by examining the Haar wavelet coefficients of the periodogram. Under normality much is known about the asymptotic distribution of the periodogram and this can be transferred, through some moderately complex distribution theory to the distribution of the Haar wavelet coefficients of the periodogram. Hence, in this situation, we have a good handle on whether a particular wavelet coefficients is too large or to small as we have near theoretical knowledge of their CDF. Since we are testing many wavelet coefficients simultaneously we have to use multiple hypothesis p-value adjustment techniques, such as Bonferroni to obtain a final p-value.

Value

An object of class `htest` containing the results of the hypothesis test. Actually a list containing the following components:

`p.val.collector`	All the p-values for all Haar wavelet coefficients of the periodogram. These are the values before p-value adjustment for multiple tests.
`p.val.adjust`	The p-values after adjustment for multiple tests via `p.adjust`.
`p.value`	The p-value of the test
`method`	Character string describing the test.

Author(s)

Delyan Savchev and Guy Nason

References

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. http://dx.doi.org/10.1002/sta4.69

See Also

`compute.rejection`, `Macdonald`

Examples

```
#
# Invent test data set which IS white noise
#
x <- rnorm(128)
#
# Do the test
#
x.wntest <- hwwn.test(x)
#
# Print the results
#
#x.wntest
#
# Wavelet Test of White Noise
#
#data:
#p-value = 0.9606
#
# So p-value indicates that there is no evidence for rejection of
# H_0: white noise.
#
# Let's do an example using data that is not white noise. E.g. AR(1)
#
x.ar <- arima.sim(n=128, model=list(ar=0.8))
#
# Do the test
#
x.ar.wntest <- hwwn.test(x.ar)
#
# Print the results
#
print(x.ar.wntest)
#
# Wavelet Test of White Noise
#
#data:
#p-value < 2.2e-16
#
```

```
# p-value is very small. Extremely strong evidence to reject H_0: white noise
```

`hywavwn.test` *Hybrid wavelet test of white noise.*

Description

Combines the general wavelet test `genwwn.test` at the medium-coarse scales and the Haar wavelet test at fine scales.

Usage

```
hywavwn.test(x, away.from = "standard", lowlev = 0, plot.it = FALSE,
stopeveryscale = FALSE, filter.number = 10,
family = "DaubExPhase", mc.method = p.adjust.methods,
verbose = FALSE, n.cdf.grid = 1000, mac.spread = 10)
```

Arguments

`x`	The time series you wish to test (of dyadic length).
`away.from`	Number of fine scales to stay away from, see details below. If "standard" then this is automatically computed for sample sizes up to length of 1024. If you have a longer series then the test will still work but might not be quite as powerful (but probably not too bad either).
`lowlev`	The coarsest coefficient to evaluate. This should always be left at 0.
`plot.it`	If TRUE then a series of plots similar to the ones produced in the `hwwn.test` function is produced. See that help page for further details on what the plots show.
`stopeveryscale`	If TRUE then if `plot.it=TRUE` then a 'scan' is issued after every plot. Just hit RETURN to continue.
`filter.number`	The number of vanishing moments of the wavelet used to compute coefficients that are then evaluated to see whether they are zero. In principle, best compression for a sparse evaluation of the normalized spectrum should mean we use the smoothest wavelets with the highest number of vanishing moments which is ten. The other components of the function are optimized for ten vanishing moments. The function will still work for other numbers of vanishing moments but maybe with slightly reduced power.
`family`	Wavelet family to go with `filter.number`.
`mc.method`	The type of multiple hypothesis correction, see `p.adjust` for details.
`verbose`	If `TRUE` some information messages are printed.
`n.cdf.grid`	The CDF of the Macdonald distribution is evaluated numerically. This argument controls the resolution of that grid: it controls the number of grid points there are between `-mac.spread` and `mac.spread`.
`mac.spread`	Horizontal range for plotting of wavelet coefficients, only used if `plot.it=TRUE`.

Details

The genwwn.test performs pretty well, but does not pick up departures from the null at the finest scale of wavelet coefficients because it does not look at those scales (because of the 'away.from' argument and the asymptotic normality that genwwn.test does not kick in at those finer scales). So, this test augments the genwwn.test with the finest scales results from hwwn.test. Those scales finer than away.from use the Haar wavelet and those coarser than away.from use the general wavelet.

Value

An object of class htest with the following components.

p.val.collector	All the of unadjusted p-values
p.val.adjust	All of the adjusted p-values
p.value	The overall p-value of the test
method	A text string describing the test
p.val.collector.hw	The of unadjusted p-values from the Haar wavelet levels
p.val.collector.gw	The of unadjusted p-values from the general wavelet levels

Author(s)

Delyan Savchev and Guy Nason

References

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. http://dx.doi.org/10.1002/sta4.69

See Also

genwwn.test, hwwn.test

Examples

```
#
# Test data
#
x <- rnorm(64)
#
# Do the test
#
answer <- hywavwn.test(x)
#
# The result in my case was:
#
#answer
```

```
#
# Hybrid Wavelet Test of White Noise
#
#data:
#p-value = 0.02305
```

`hywn.test` *Hybrid of Box-Ljung test, Bartlett B test, Haar wavelet and General wavelet tests.*

Description

Omnibus test that attempts to mitigate poor performance of single test on a particular class where it does poorly by running four different tests that work well in different directions and pooling their results.

Usage

```
hywn.test(x, filter.number = 10, family = "DaubExPhase")
```

Arguments

`x`	The data you wish to test (dyadic length)
`filter.number`	The number of vanishing moments of the wavelet used.
`family`	The family of the wavelets

Value

A list with components:

`p.value`	The overall p-value of the test
`method`	Text string containing the name of the method used

Author(s)

Delyan Savchev and Guy Nason

References

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. http://dx.doi.org/10.1002/sta4.69

See Also

`bartlettB.test`, `genwwn.test`, `hwwn.test`

Examples

```
#
# Generate test data
#
x <- rnorm(64)
#
# Run the hybrid test
#
hywn.test(x)
#
# Hybrid Test
#
#data:
#p-value = 0.09221
```

`Macdonald` *Compute the Macdonald density function for a specified parameter value* `m` *at a vector of* `x` *values.*

Description

Compute the Macdonald density function for a specified parameter value `m` at a vector of `x` values.

Usage

```
Macdonald(x, m)
```

Arguments

`x`	The `x` ordinates that you want to evaluate the density at. A vector of real numbers.
`m`	The parameter of the Macdonald's density

Details

This function computes the Macdonald probability density function for parameter `m` and values at which to evaluate the density supplied in `x`. The mean and variance of this density is zero and one respectively.

Value

The density

Author(s)

Delyan Savchev and Guy Nason

References

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. http://dx.doi.org/10.1002/sta4.69

See Also

compute.rejection, hwwn.test

Examples

```
#
# Work out density at x=0, 0.5 and 1 for the m=1 Macdonald density
#
Macdonald(x=c(0,0.5,1), m=2)
#[1] 0.3535534 0.2975933 0.2075131
#
# Check that the density integrates to one, e.g. for m=3
#
integrate(Macdonald, lower=-20, upper=20, m=3)
#1 with absolute error < 4.7e-07
```

---

sqcoefvec *Compute coefficients required for approximaing the wavelet transform using the square of wavelets.*

Description

Essentially, part of a method for computing a wavelet-like transform using the squares of wavelets rather than the wavelets themselves.

Usage

```
sqcoefvec(m0, filter.number = 10, family = "DaubLeAsymm",
resolution = 4096, stop.on.error = FALSE, plot.it = FALSE)
```

Arguments

`m0`	The number of scales finer than the square wavelet being approximated. Usually, 2 or 3 is enough.
`filter.number`	Number of vanishing moments of underlying wavelet.
`family`	Family of underlying wavelet
`resolution`	Function values of the wavelet itself are generated by a high-resolution approximation. This argument specifies exactly how many values.
`stop.on.error`	This argument is supplied to the `integrate` function which performs numerical integration within this code.
`plot.it`	Plots showing the approximation are plotted.

Details

The idea is that the square of a wavelet (the square wavelet) is approximated by wavelets at a finer scale. The argument `m0` controls how many levels below the original scale are used. Essentially, this function computes a representation of the original square wavelet in terms of finer scale wavelets. Hence, when a decomposition of another function with respect to the square wavelets is required, one can compute the representation with respect to a regular wavelet decomposition and then apply the wavelet to square wavelet transform to turn it into a square wavelet representation.

This idea originally used for performing 'powers of wavelets' transforms in Herrick (2000) and Barber, Nason and Silverman (2002) and for the mod-wavelets is described in Fryzlewicz, Nason and von Sachs (2008).

Value

A list with the following components:

`ll`	Vector containing integers between the lower and upper limit of the wavelets required at the finer scale.
`ecoef`	The appropriate coefficients that approximate the mod wavelet at the finer scale.
`m0`	The number of scales finer below the scale that the function is at
`filter.number`	The wavelet filter number used
`family`	The wavelet family used
`ecode`	An error code, if zero then ok, otherwise returns 1
`ians`	The actual return values from the internal call to the `integrate` function

Author(s)

Guy Nason

References

Barber, S., Nason, G.P. and Silverman, B.W. (2002) Posterior probability intervals for wavelet thresholding. *J. R. Statist. Soc. B*, **64**, 189-206.

Fryzlewicz, P., Nason, G.P. and von Sachs, R. (2008) A wavelet-Fisz approach to spectrum estimation. *J. Time Ser. Anal.*, **29**, 868-880.

Herrick, D.R.M. (2000) *Wavelet Methods for Curve Estimation*, PhD thesis, University of Bristol, U.K.

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. `http://dx.doi.org/10.1002/sta4.69`

See Also

`sqwd`, `sqndwd`, `sqndwdecomp`

Examples

```
#
# This function is not really designed to be used by the casual user
#
tmp <- sqcoefvec(m0=2, filter.number=4)
```

`sqndwd` *Compute the non-decimated squared wavelet transform.*

Description

A transform of the data with respect to an expansion comprised of squared wavelets.

Usage

```
sqndwd(x, ec)
```

Arguments

`x`	The sequence that you want transformed, of dyadic length.
`ec`	A structure containing the necessary information to transform the wavelet transform of the sequence to the squared wavelet transform. This is provided by the function `sqcoefvec`.

Details

This function first computes the wavelet transform of the `x` data. Then, level by level it is retransformed into the coefficients of the squared-wavelet transform using the `ec` structure. Fine levels, that cannot be computed using the approximate method are computed directly by the brute-force method in `sqndwdecomp`. Method used is described in Fryzlewicz, Nason and von Sachs (2008), and is analogous to the 'powers of wavelets' transform described in Herrick (2000) and Barber, Nason and Silverman (2002).

Value

An object of class `wd` containing the non-decimated squared wavelet transform.

Author(s)

Guy Nason

References

Barber, S., Nason, G.P. and Silverman, B.W. (2002) Posterior probability intervals for wavelet thresholding. *J. R. Statist. Soc. B*, **64**, 189-206.

Fryzlewicz, P., Nason, G.P. and von Sachs, R. (2008) A wavelet-Fisz approach to spectrum estimation. *J. Time Ser. Anal.*, **29**, 868-880.

Herrick, D.R.M. (2000) *Wavelet Methods for Curve Estimation*, PhD thesis, University of Bristol, U.K.

Nason, G.P. and Savchev, D. (2014) White noise testing using wavelets. *Stat*, **3**, 351-362. http://dx.doi.org/10.1002/sta4.69

See Also

sqwd

Examples

```
#
# Random data
#
x <- rnorm(16)
#
# Compute the projection of x onto the squared wavelets...
#
ans <- sqndwd(x, ec=sqcoefvec(m0=2, filter.number=4))
```

---

sqndwdecomp *Brute-force calculation of the non-decimated squared wavelet transform.*

Description

Accurate, but brute-force, direct (slow) calculation of the non-decimated squared wavelet transform.

Usage

```
sqndwdecomp(x, J, filter.number, family)
```

Arguments

x	The sequence you want to transform
J	The number of resolutions you want
filter.number	The wavelet filter you wish to use.
family	The wavelet family you wish to use

Details

Works by computing the discrete wavelets and the necessary scales using `hwwn.dw` function. Then forms the direct inner product with the data with the squares of the discrete wavelets.

Value

Returns a matrix of J rows and length(x) columns. Row j in the matrix returned corresponds to the level (nlev-j) resolution level coefficients (where nlev is nlevelsWT(x)) in the WaveThresh ordering.

Author(s)

Piotr Fryzlewicz (modified by Guy Nason)

References

Fryzlewicz, P., Nason, G.P. and von Sachs, R. (2008) A wavelet-Fisz approach to spectrum estimation. *J. Time Ser. Anal.*, **29**, 868-880.

See Also

`sqndwd`

Examples

```
#
# Generate random series and then take transform
x <- rnorm(128)
y <- sqndwdecomp(x=x, J=2, filter.number=3, family="DaubExPhase")
```

---

sqwd *Compute expansion with respect to squared wavelets.*

Description

Compute expansion with respect to squared wavelets. See help for `sqndwd`. The coefficients are the projection of the input sequence onto the set of functions being the squares of the usual wavelets. This operation is most useful for computing variances of wavelet coefficients.

Usage

```
sqwd(x, filter.number = 10, family = "DaubExPhase", type = "station", m0 = 3)
```

Arguments

`x`	Sequence that you wish to compute expansion for.
`filter.number`	Base wavelet family (no. of vanishing moments) you wish to use.
`family`	The base wavelet family you wish to use.
`type`	Either `station` for the non-decimated transform or `wavelet` for the regular wavelet transform.
`m0`	The number of scales down (finer) from the scale of the squared wavelet being approximated. Usually, 2 or 3 is enough. Many more scales results in a better approximation but at a higher cost as the number of coefficients at consecutive scales doubles.

Details

This function is an implementation of the 'powers of wavelets' idea from Herrick (2000), Barber, Nason and Silverman (2002) and, for the associated mod-wavelets by Fryzlewicz, Nason and von Sachs (2008).

Value

An object of class `wd` but containing coefficients with respect to the squared wavelets.

Author(s)

Guy Nason

References

Barber, S., Nason, G.P. and Silverman, B.W. (2002) Posterior probability intervals for wavelet thresholding. *J. R. Statist. Soc. B*, **64**, 189-206.

Fryzlewicz, P., Nason, G.P. and von Sachs, R. (2008) A wavelet-Fisz approach to spectrum estimation. *J. Time Ser. Anal.*, **29**, 868-880.

Herrick, D.R.M. (2000) *Wavelet Methods for Curve Estimation*, PhD thesis, University of Bristol, U.K.

See Also

`genwwn.thpower`, `sqcoefvec`, `sqndwd`, `sqndwdecomp`

Examples

```
#
# A made-up sequence
#
x <- 1:32
#
# Work out its expansion wrt squared wavelets
#
x.sqwd <- sqwd(1:32)
```

Index