Package: BootWPTOS (via r-universe)

June 9, 2026

Title Test Stationarity using Bootstrap Wavelet Packet Tests

Version 1.2.1

Date 2022-05-19

Depends R (>= 2.0), wavethresh, tseries

Description Provides significance tests for second-order stationarity for time series using bootstrap wavelet packet tests. Provides functionality to visualize the time series with the results of the hypothesis tests superimposed. The methodology is described in Cardinali, A and Nason, G P (2016) ``Practical powerful wavelet packet tests for second-order stationarity." Applied and Computational Harmonic Analysis, 44, 558-585 <doi:10.1016/j.acha.2016.06.006>.

License GPL-2

NeedsCompilation no

Author Guy Nason [aut, cre], Alessandro Cardinali [aut]

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

Config/pak/sysreqs libssl-dev

Repository https://guynason.r-universe.dev

Date/Publication 2022-05-20 11:40:02 UTC

RemoteUrl https://github.com/cran/BootWPTOS

RemoteRef HEAD

RemoteSha 345d143c842b942dcaffcfaf4edf4fdbaeeb40c6

Contents

`BootWPTOS-package` *Bootstrap Wavelet Packet Tests of Stationarity*

Description

This package contains two main functions to carry out tests of second-order stationarity using wavelet packets. One test, `BootWPTOS` carries out the bootstrap wavelet packet test of stationarity as described by Cardinali and Nason (2016), with algorithm of that same name in that paper. The test is carried out respect to a set of wavelet packets (one, or more than one).

The other main function is `WPTOSpickout`. Here, the test is carried out using a fixed single wavelet packet and inference for test statistics is carried out using asymptotic normal approximations as described in Cardinali and Nason (2016) but based on ideas in von Sachs and Neumann (2000).

Details

Package:	BootWPTOS
Type:	Package
Version:	1.2.1
Date:	2022-05-19
License:	GPL-2

The main functions are documented above. See below for an example of each.

Both functions require the specification of a set (for `BootWPTOS`) or a single (for `WPTOSpickout`) wavelet packet. This is because the tests use and rely on wavelet packets.

Wavelet packets are indexed by two quantities: scale and index. The scale is referred to in the functions by the `levs` and `level` arguments respectively. Scale can be any scale that you would normally use in wavethresh. So, for a series of dyadic length, that is T=2^J, the scales are indexed 0 (coarsest scale) to J-1 (finest scale).

The range of the index argument for a wavelet packet depends on the scale. Always the scaling function coefficients have index 0 and the regular wavelet coefficients always have index 1. Note: wavelets are a subset of wavelet packets. Then, for scale J-j there are 2^j packets.

So, for example, at the finest scale J-1 there are 2^1=2 packets. These correspond to indices 0 and 1, the father and mother wavelet coefficients respectively. For the next finest scale J-2 there are 2^2=4 packets. These are indexed 0, 1, 2 and 3 with 0,1 being the father/mother wavelet (packet) coefficients at that scale and indices 2 and 3 being the other two wavelet packets at that scale. Clearly there are many more wavelet packets at coarser scales.

In our functions the levs or level contains the scale of any wavelet packet and the indices or index variable contains the indices.

In `BootWPTOS` you can use any combination of wavelet packets, but it is important that the entries correspond to each other in the levs and indices vector. E.g. if you wanted wavelet packet (3,5) and (4,7) with J=5 then you would use the arguments levs=c(3,4) and indices=c(5,7).

Author(s)

Guy Nason

Maintainer: Who to complain to <g.nason@imperial.ac.uk>

References

Cardinali, A. and Nason, G.P. (2016) Practical Powerful Wavelet Packet Tests for Second-Order Stationarity. Applied and Computational Harmonic Analysis, 2016, doi:10.1016/j.acha.2016.06.006

Von Sachs, R. and Neumann, M.H. (2000) A Wavelet-Based Test for Stationarity. Journal of Time Series Analysis, 21, 597-613. doi:10.1111/14679892.00200

See Also

BootWPTOS, plot.toswp, print.toswp, summary.toswp, WPTOSpickout

Examples

```
#
# First, we provide an example concerning BootWPTOS
#
#
# Generate a stationary time series (e.g. iid standard normals)
#
x <- rnorm(512)
#
# What would be the finest scale?
#
J <- IsPowerOfTwo(length(x))
J
#[1] 9
#
# So, in WaveThresh there are 9 scaled indexed 0 to 8.
#
# Test x for stationarity
#
# The finest scale wavelets are at 8
# The next finest scale is 7.
#
# Wavelets themselves are always indexed 1, father wavelets 0.
# We don't tend to use father wavelets for stationary testing.
#
# There are 2^j packets at scale J-j (so 2 at the finest [father and
# mother], 4 at the next finest [father=0, mother=1, packets 2 and 3].
#
# Let's just look at the finest scale wavelets and the next finest
# scale wavelets and two other wavelet packets.
#
x.test <- BootWPTOS(x=x, levs=c(8,7,7,7), indices=c(1,1,2,3), Bsims=30)
#
# Note: Bsims=30 is ALMOST CERTAINLY TOO SMALL (but it is small here because
# on installation R run these examples and I don't want it to take too long.
```

```
# 100+ is almost certainly necessary, and probably 500+ useful and 1000+
# to be "sure". If you can load the multicore library then you can
# replace lapplyfn=lapply with lapplyfn=mclapply to get a parallel processing
#
# What are the results of our test?
#
x.test
#
#       WPBootTOS test of stationarity
#
#data:  x
#= 1.8096, p-value = 0.7
#
# So, the p-value is > 0.05 so this test indicates that there is
# no evidence for non-stationarity. Running it for 1000 bootstrap simulations
# gave a p-vale of 0.736.
#
# The next example is nonstationary. However, after the series has been
# generated you should plot it. The second half has a different variance
# to the first half but it is very difficult, usually, to identify the
# different variances on a plot.
#
x2 <- c(rnorm(256), rnorm(256,sd=1.4))
#
# Let's do a test, but involve ALL non-father-wavelet packets from scales
# 8, 7, 6 and 5.
#
# NOTE: Typically we use much more than 50 bootstrap sims
#
#
x2.test <- BootWPTOS(x=x2, levs=c(8,7,7,7,rep(6,7), rep(5,15)),
 indices=c(1,1,2,3, 1:7, 1:15), Bsims=30)
x2.test
#
#       WPBootTOS test of stationarity
#
#data:  x2
#= 5.4362, p-value < 2.2e-16
#
# So, strong evidence for nonstationarity because p.value < 0.05 (much less
# than!).
# Using Bsims=1000 and mclapply (for speed) gave a p-value
# of 0.002, so still assessed to be nonstationary, but we have more confidence
# in the answer.
#
# Now we provide an example of using WPTOSpickout
#
#
# Create some simulated data
#
x2 <- c(rnorm(256), rnorm(256, 2))
#
# Note: x2 should be highly nonstationary. The left-hand half of the series
```

```
# has variance 1, the right-hand half has variance 2.
#
# First, try out a wavelet packet test of stationarity. This is a check
# on the later test. You should really check both.
#
# We've chosen this packet more or less at random. Its packet at scale 5 and
# index 1 (this happens to be a wavelet)
#
# Again, typically we use more than 30 bootstrap sims, 30 is too small
#
x2.tos <- BootWPTOS(x2, levs=5, indices=1, Bsims=30)
x2.tos
#
#       WPBootTOS test of stationarity
#
#data:  x2
#= 5.2826, p-value < 2.2e-16
#
# So, test indicates that strong evidence for nonstationarity.
#
# Now let's do the multiple Haar hypothesis test.
#
x2.po <- WPTOSpickout(x=x2, level=7, index=1)
x2.po
#Class 'toswp' : Wavelet Packet Test of Stationarity Object :
#       ~~~~  : List with 7 components with names
#               x level index sigcoefs nreject ntests bonsize
#
#
#summary(.):
#----------
#Number of individual tests: 56
#Bonferroni p-value was: 0.0008928571
#Tests rejected: 2
#Listing Bonferroni rejects...
#Wavelet Packet  (5,1): HWTlev: 4. Indices: 8
#Wavelet Packet  (5,1): HWTlev: 5. Indices: 16
#
# So, this test also shows nonstationarities. For this packet (5,1)
# two significant Haar coefficients were identified. One was at level 4
# position 8 and the other was at scale level 5 position 16.
#
# You can plot them also
#
plot(x2.po)
#
# You should get a nice plot of the time series with double-headed red
# arrows indicating the location and extent of the nonstationarities.
# For this example, where the spectrum changes dramatically at the halfway
# point - this is where the arrows should be located. Of course, with random
# data you might see other arrows in other locations, but this should be
# unlikely and on repeating the above they should not persist.
```

BootWPTOS *Compute test of stationarity for time series via bootstrap wavelet packet method.*

Description

Computes b-spectrum (wavelet packet periodogram) for predefined set of wavelet packets on time series. Then applies bootstrap method to resample new versions of the (assumed for the test) stationary series and retests the series. If the value of the test statistic is out of line (bigger) than the resampled test statistics then the series might well not be stationary.

Usage

```
BootWPTOS(x, levs, indices, filter.number = 1, family = "DaubExPhase",
 Bsims = 200, lapplyfn = lapply, ret.all = FALSE)
```

Arguments

x	The time series you wish to test. Length is a power of two
levs	The levels of the wavelet packets that you want to involve in the test.
indices	The indices of the wavelet packets that you want to involve in the test.
filter.number	The filter number of the wavelet that underlies the wavelet packet.
family	The family of the wavelet that underlies the wavelet packet used for the test.
Bsims	The number of bootstrap simulations.
lapplyfn	By default this argument is the lapply function which operates sequentially. However, if you have the **parallel** package you could supply the mclapply function which processes list elements in parallel on a multicore machines.
ret.all	If FALSE then the results of the test are returned in the standard R htest object, and can be printed and manipulated by those standard tools. If TRUE then a list is returned with information about the test.

Details

Function computes a test statistic for test of stationarity on a time series. Then successive bootstrap realizations are drawn using the surrogate function. If the original time series WAS stationary then surrogate causes stationary draws with the same spectral characteristic as the data to be produced (with Gaussian marginals). Under the null hypothesis the time series is assumed stationary and so the distribution of all of the test statistics should be the same and the p-value of the test statistic be uniformly distributed. If the series is nonstationary then the value of the statistic is likely to be bigger on the first computed test statistic on the dats and much bigger than all the others. We can work out a bootstrap p-value by counting how many resampled test statistics are bigger than the one computed on the data.

Value

Normally a list, cast as a `htest` class object with the following components:

`statistic`	The test statistic computed on the data
`p.value`	The bootstrap p.value of the test.
`method`	The name of the method of this test statistic.
`data.name`	The name of the data set tested.
`Bootvals`	The remaining bootstrap generated test statistics.

Author(s)

G.P. Nason

References

Cardinali, A. and Nason, G.P. (2016) Practical Powerful Wavelet Packet Tests for Second-Order Stationarity. Applied and Computational Harmonic Analysis, 2016 doi:10.1016/j.acha.2016.06.006

See Also

`WPts`, `WPTOSpickout`

Examples

```
#
# Generate a stationary time series (e.g. iid standard normals)
#
x <- rnorm(512)
#
# What would be the finest scale?
#
J <- IsPowerOfTwo(length(x))
J
#[1] 9
#
# So, in WaveThresh there are 9 scales indexed 0 to 8.
#
# Let us test x for stationarity
#
# The finest scale wavelets (or packets) are at scale 8
# The next finest scale is 7.
#
# Wavelets themselves are always indexed 1, father wavelets 0.
# We don't tend to use father wavelets for stationary testing.
#
# There are 2^j packets at scale J-j (so 2 at the finest [father and
# mother], 4 at the next finest [father=0, mother=1, packets 2 and 3].
#
# Let's just look at the finest scale wavelet (8,1) and the next finest
# scale wavelet (7,1) and two other wavelet packets  (7,2) and (7,3)
```

```
#
x.test <- BootWPTOS(x=x, levs=c(8,7,7,7), indices=c(1,1,2,3), Bsims=30)
#
# Note: Bsims=30 is almost certainly too small (but it is small here because
# on installation R run these examples and I don't want it to take too long.
# 100+ is almost certainly necessary, and probably 500+ useful and 1000+
# to be "sure". If you can load the multicore library then you can
# replace lapplyfn=lapply with lapplyfn=mclapply to get a parallel processing
#
# What are the results of our test?
#
x.test
#
# WPBootTOS test of stationarity
#
#data:  x
#= 1.8096, p-value = 0.7
#
# So, the p-value is > 0.05 so this test indicates that there is
# no evidence for non-stationarity. Running it for 1000 bootstrap simulations
# gave a p-vale of 0.736.
#
# The next example is nonstationary. However, after the series has been
# generated you should plot it. The second half has a different variance
# to the first half but it is very difficult, usually, to identify the
# different variances on a plot.
#
x2 <- c(rnorm(256), rnorm(256,sd=1.4))
#
# Let's do a test, but involve ALL non-father-wavelet packets from scales
# 8, 7, 6 and 5.
#
x2.test <- BootWPTOS(x=x2, levs=c(8,7,7,7,rep(6,7), rep(5,15)),
 indices=c(1,1,2,3, 1:7, 1:15), Bsims=30)
x2.test
#
# WPBootTOS test of stationarity
#
#data:  x2
#= 5.4362, p-value < 2.2e-16
#
# So, strong evidence for nonstationarity because p.value < 0.05 (much less
# than!). Again here we've only use 30 bootstrap simulations and this is
# probably too small. Using Bsims=1000 and mclapply (for speed) gave a p-value
# of 0.002, so still assessed to be nonstationary, but we have more confidence
# in the answer.
```

`plot.toswp` *Plot* `toswp` *class object.*

Description

Plots the time series that was analyzed to produce the toswp class object. Then superimposes the location and extent of nonstationarities by means of double-headed red arrows. The right-hand axis indicates the scale of the significant Haar wavelet coefficients corresponding to the nonstationary arrows.

Usage

```
## S3 method for class 'toswp'
plot(x, sub = NULL, xlab = "Time", arrow.length = 0.05, verbose = FALSE, ...)
```

Arguments

x	The toswp class object that you wish to plot.
sub	A subtitle.
xlab	The label for the x-axis.
arrow.length	Length of arrow head.
verbose	If TRUE prints debugging information from the plotting process.
...	Other arguments to the plot function.

Details

As description says.

Value

Nothing of interest.

Author(s)

G.P. Nason

References

Cardinali, A. and Nason, G.P. (2016) Practical Powerful Wavelet Packet Tests for Second-Order Stationarity. Applied and Computational Harmonic Analysis, 2016. doi:10.1016/j.acha.2016.06.006

See Also

WPTOSpickout, print.toswp, summary.toswp

Examples

```
#
# See example in helpfile for \code{\link{WPTOSpickout}}
#
```

print.toswp *Print* toswp *class object.*

Description

Prints introduction to toswp class object.

Usage

```
## S3 method for class 'toswp'
print(x, ...)
```

Arguments

x	The print.toswp object that you wish to print
...	Other arguments to print

Details

Just prints out some basic facts about the object.

Value

Nothing!

Author(s)

G.P.Nason

References

Cardinali, A. and Nason, G.P. (2016) Practical Powerful Wavelet Packet Tests for Second-Order Stationarity. Applied and Computational Harmonic Analysis, 2016. doi:10.1016/j.acha.2016.06.006

See Also

WPTOSpickout, plot.toswp, summary.toswp

Examples

```
#
# See example in \code{\link{WPTOSpickout}}
```

summary.toswp *Summarize a* toswp *class object.*

Description

This function goes through a toswp class object and printing out details of which Haar coefficients were significant.

Usage

```
## S3 method for class 'toswp'
summary(object, quiet = FALSE, ...)
```

Arguments

object	The object you wish to summarize.
quiet	If TRUE nothing is printed. However, a list of the significant coefficients is returned. This information is used, for example, by plot.toswp.
...	Other arguments to summary.

Details

None

Value

A list with the following components:

rejlist	A list with details on the rejected coefficients. Each component of the list is a vector. The first element of each vector is the Haar wavelet coefficient scale level. The remaining numbers are the indices of any significant coefficients at that level.
nreject	The number of rejected coefficients

Author(s)

G.P. Nason

References

Cardinali, A. and Nason, G.P. (2016) Practical Powerful Wavelet Packet Tests for Second-Order Stationarity. Applied and Computational Harmonic Analysis, 2016. doi:10.1016/j.acha.2016.06.006

See Also

WPTOSpickout, plot.toswp, print.toswp

Examples

```
#
# See example of \code{\link{print.toswp}} in the help for
# \code{\link{WPTOSpickout}} which includes a call to this function.
```

`WPTOSpickout` *Find nonstationarities in a time series*

Description

The nonstationarities are located by looking for significant Haar wavelet coefficients of a b-spectrum of a time series (a b-spectrum is the expectation of a wavelet packet periodgram). The significant Haar coefficients can locate discontinuities in space and time.

Usage

```
WPTOSpickout(x, level, index, filter.number = 1, family = "DaubExPhase",
 plot.it = FALSE, verbose = FALSE, lowlev = 3, highlev, nomsize = 0.05)
```

Arguments

`x`	The time series you wish to analyse.
`level`	The level of the b-spectrum you want to examine. See help for `BootWPTOS` for more information on levels.
`index`	The index of the b-spectrum you want to examine. See help for `BootWPTOS` for more information on indices you can chose.
`filter.number`	The filter number of the underlying wavelet you wish to examine.
`family`	The family of the underlying wavelet you wish to examine.
`plot.it`	If `TRUE` this plots the time series `x` as supplied. Then, superimposed in red, is the raw wavelet packet periodogram. The Haar wavelet coefficients of the red signal are examined for whether they are deemed non-zero or not using asymptotic normality results. If `FALSE` then no such plot is produced.
`verbose`	If `TRUE` then a single line indicating the number of significant coefficients found is printed. If `FALSE` then the function prints nothing.
`lowlev`	Keep away from coarse scales. Typically, Haar wavelet coefficients at the coarse scales are contaminated by boundary effects. These won't usually cause a problem at scales 3 or higher, or maybe 2. Only coefficients at scales `lowlev` or finer will be selected for testing.
`highlev`	Keep away from fine scales. The testing of Haar wavelet coeffients depends on utilizing enough data points to enable asymptotic normality to kick in. At the finest scale only TWO single points are compared and the distribution of each point might be far from normality. At coarser scales TWO averages are compared and those averages will consist of many points. E.g. at the third finest scale each average will consist of four points, at the fourth finest scale

each average will consist of 8 points, etc. By default this argument is set to be roughly the use a scale one level finer than the halfway number of levels. So, if J=10, then highlev is 6. The formula is floor(J/2)+1. Note: highlev and lowlev should be specified in the WaveThresh scaling (e.g. scale 0 is the coarsest scale)

`nomsize` The nominal size of the test as a number between 0 and 1. So, if you want a 5

Details

This function computes the nondecimated wavelet packet transform of the packet you specified by `level` and `index`. Note: you can only specify one number for each of these. Then the b-spectrum (raw wavelet packet periodogram) is formed by squaring the nondecimated wavelet packet transform. Then the Haar wavelet coefficients are obtained for the b-spectrum and a multiple hypothesis test is performed on all the Haar wavelet coefficients between scales lowlev and highlev. The function return information about any significant wavelet coefficients.

Value

A list of class `toswp` containing the following components:

`x`	The time series that was analyzed
`level`	The level of the b-spectrum that we wanted
`index`	The index of the b-spectrum that we wanted
`sigcoefs`	A wd class object containing the significant Haar wavelet coefficients, if there are any
`nreject`	The number of significant Haar wavelet coefficients
`ntests`	The total number of hypothesis tests carried out in the multiple hypothesis test
`bonsize`	The Bonferroni corrected rate for the multiple hypothesis test

Author(s)

G.P. Nason

References

Cardinali, A. and Nason, G.P. (2016) Practical Powerful Wavelet Packet Tests for Second-Order Stationarity. Applied and Computational Harmonic Analysis, 2016 doi:10.1016/j.acha.2016.06.006

See Also

`BootWPTOS`, `plot.toswp`, `print.toswp`, `summary.toswp`

Examples

```
#
# Create some simulated data
#
x2 <- c(rnorm(256), rnorm(256, 2))
#
```

```
# The following call to BootWPTOS (generic tester)
#
# [We're not running them in R package testing as they can be quite intensive]
#
## Not run: x2.tos <- BootWPTOS(x2, levs=5, indices=1, Bsims=500)
## Not run: x2.tos
#
# WPBootTOS test of stationarity
#
#data:  x2
#= 5.2826, p-value < 2.2e-16
#
# So, test indicates that strong evidence for nonstationarity.
#
# Now let's do the multiple Haar hypothesis test.
#
x2.po <- WPTOSpickout(x=x2, level=7, index=1)
x2.po
#Class 'toswp' : Wavelet Packet Test of Stationarity Object :
#       ~~~~  : List with 7 components with names
#               x level index sigcoefs nreject ntests bonsize
#
#
#summary(.):
#----------
#Number of individual tests: 56
#Bonferroni p-value was: 0.0008928571
#Tests rejected: 2
#Listing Bonferroni rejects...
#Wavelet Packet  (5,1): HWTlev: 4. Indices: 8
#Wavelet Packet  (5,1): HWTlev: 5. Indices: 16
#
# So, this test also shows nonstationarities. For this packet (5,1)
# two significant Haar coefficients were identified. One was at level 4
# position 8 and the other was at scale level 5 position 16.
#
# You can plot them also
#
plot(x2.po)
#
# You should get a nice plot of the time series with double-headed red
# arrows indicating the location and extent of the nonstationarities.
# For this example, where the spectrum changes dramatically at the halfway
# point - this is where the arrows should be located. Of course, with random
# data you might see other arrows in other locations, but this should be
# unlikely and on repeating the above they should not persist.
```

WPts *Compute wavelet packet test statistic for test of stationarity.*

Description

Computes nondecimated wavelet packet transform of time series. Computes b-spectrum (square of nondecimated WP transform) for various levels and indices (controlled by `levs` and `indices` arguments). Computes variance (L2 norm) of the b-spectra and averages them. Returns the average.

Usage

```
WPts(x, levs, indices, filter.number = 1, family = "DaubExPhase")
```

Arguments

`x`	The time series whose statistic you want to compute.
`levs`	The b-spectrum levels you want to use.
`indices`	The b-spectrum indices you want to use.
`filter.number`	The filter number of the underlying wavelet.
`family`	The family of the underlying wavelet.

Details

Description says it all. However, the `levs` and `indices` warrant further explanation.

Our code is designed to be used on data sets that are a power of two, i.e. T = 2^J for some J (note: the test can work on other values of T but coding is more finickity). Given a series of this length there are J levels, labelled 0 (coarse) to J-1 fine. Within each level there are J-j packets indexed 0, 1, ..., J-j-1 for scales J-1, ..., 0 respectively.

Packet 0 within any scale always corresponds to the father wavelets at that scale and we don't tend to use this for stationarity testing. Packet 1 within any scale always corresponds to mother wavelets. We often use these. Note, at the finest scale J-1 there are only two packets 0 (father wavelet) and 1 (mother wavelet) coefficients.

Value

The test statistic value is returned.

Author(s)

G.P. Nason

References

Cardinali, A. and Nason, G.P. (2016) Practical Powerful Wavelet Packet Tests for Second-Order Stationarity. Applied and Computational Harmonic Analysis, 2016. doi:10.1016/j.acha.2016.06.006

See Also

`BootWPTOS`

Examples

```
#
# Generate some test data
#
x <- rnorm(512)
#
# Compute the test statistic on mother wavelets and packets from the finest
# scale and the THIRD finest scale
#
J <- IsPowerOfTwo(length(x))
J
# [1] 9
#
x.ts <- WPts(x, levs=c(8, rep(6, 7)), indices=c(1, 1:7))
x.ts
# [1] 1.792252
```

Index