Package 'wbsts'

August 23, 2015

Title Multiple Change-Point Detection for Nonstationary Time Series

Version 0.1

Author Karolos Korkas and Piotr Fryzlewicz

Maintainer Karolos Korkas <kkorkas@yahoo.co.uk>

Description Implements detection for the number and locations of the change-points in a time series using the Wild Binary Segmentation and the Locally Stationary Wavelet model.

Depends mvtnorm,wmtsa, R (>= 3.0.0)

License GPL-3

LazyData true

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-08-23 23:06:37

R topics documented:

wbsts-package *Multiple change-point detection for nonstationary time series*

Description

Implements the Wild Binary Segmentation method of Fryzlewicz (2014) for nostationary time series as described in Korkas and Fryzlewicz (2015). Its purpose is the estimation of the number and locations of the change-points in a time series utilising the wavelet periodogram.

Author(s)

K. Korkas and P. Fryzlewicz

References

P. Fryzlewicz (2014), Wild Binary Segmentation for multiple change-point detection. Annals of Statistics, 42, 2243-2281. (http://stats.lse.ac.uk/fryzlewicz/wbs/wbs.pdf)

K. Korkas and P. Fryzlewicz (2015), Multiple change-point detection for non-stationary time series using Wild Binary Segmentation. (http://stats.lse.ac.uk/fryzlewicz/WBS_LSW/WBS_LSW.pdf)

Examples

```
#### Generate a highly persistent time series with changing variance and of length 5,000
###Location of the change-points
cps=seq(from=1000,to=2800,by=200)
y=sim.pw.arma(N =3000,sd_u = c(1,1.5,1,1.5,1,1.5,1,1.5,1,1.5,1),
b.slope=rep(0.99,11),b.slope2 = rep(0.,11), mac = rep(0.,11),br.loc = cps)[[2]]
###Estimate the change points via Binary Segmentation
wbs.lsw(y,M=1)$cp.aft
###Estimate the change points via Wild Binary Segmentation
wbs.lsw(y,M=0)$cp.aft
```

cr.rand.max.inner.prod *The value that maximises the random CUSUM statistic across all the scales*

Description

The function finds the value which yields the maximum inner product with the input time series (CUSUM) located between $100(1-p)\%$ and $100p\%$ of their support across all the wavelet periodogram scales.

Usage

```
cr.rand.max.inner.prod(XX,Ts,C_i,epp,M = 0,Plot = FALSE,cstar=0.95)
```

Arguments

XX	The wavelet periodogram.
Ts	The sample size of the series.
C_i	The CUSUM threshold.
epp	A minimum adjustment for the bias present in $E_{t,T}^{(i)}$.
M	Number of random CUSUM to be generated.
Plot	Plot the threhsold CUSUM statistics across the wavelet scales.
cstar	A scalar in (0.67,1]

Value

1	Candidate change point
2	The maximum CUSUM value
3	The starting point s of the favourable draw
4	The ending point e of the favourable draw

Author(s)

K. Korkas and P. Fryzlewicz

References

K. Korkas and P. Fryzlewicz (2015), Multiple change-point detection for non-stationary time series using Wild Binary Segmentation. (http://stats.lse.ac.uk/fryzlewicz/WBS_LSW/WBS_LSW.pdf)

Examples

```
cps=seq(from=1000,to=2000,by=200)
y=sim.pw.arma(N =3000,sd_u = c(1,1.5,1,1.5,1,1.5,1),
b.slope=rep(0.99,7),b.slope2 = rep(0.,7), mac = rep(0.,7),br.loc = cps)[[2]]
z=ews.trans(y,scales=c(11,9,8,7,6))
out=cr.rand.max.inner.prod(z, Ts = length(y),C_i = tau.fun(y),
epp = rep(32,5), M = 2000, cstar = 0.75, Plot = 1)
abline(v=cps,col="red")
```

---

ews.trans *Computation of the Evolutionary Wavelet Spectrum (EWS)*

Description

The function computes the EWS from a time series of any (non-dyadic) size by utilising the maximal overlap discrete wavelet transform; see also W. Constantine and D. Percival (2015).

Usage

```
ews.trans(x,scales=NULL)
```

Arguments

x	The time series.
scales	The wavelet periodogram scales to compute starting from the finest.

Value

The evolutionary wavelet spectral estimate of y.

References

W. Constantine and D. Percival (2015), wmtsa: Wavelet Methods for Time Series Analysis. R Package Version 2.0-0.

Examples

```
ews=ews.trans(rnorm(1000),c(9,8,7))
barplot(ews[,1])
```

---

finner.prod.iter *A fast implementation of the CUSUM statistic*

Description

The function returns the CUSUM statistic of a time series by calling the C function fast_inner_prod

Usage

```
finner.prod.iter(x)
```

Arguments

x	A time series

Author(s)

K. Korkas and P. Fryzlewicz

References

K. Korkas and P. Fryzlewicz (2015), Multiple change-point detection for non-stationary time series using Wild Binary Segmentation. (http://stats.lse.ac.uk/fryzlewicz/WBS_LSW/WBS_LSW.pdf)

Examples

```
cps=seq(from=1000,to=2000,by=200)
y=sim.pw.arma(N =3000,sd_u = c(1,1.5,1,1.5,1,1.5,1),
b.slope=rep(0.99,7),b.slope2 = rep(0.,7), mac = rep(0.,7),br.loc = cps)[[2]]
z=ews.trans(y,scales=c(11,9,8,7,6))
ts.plot(abs(finner.prod.iter(z[10:2990,2])))
```

`get.thres` *Universal thresholds calculation*

Description

The function returns universal thresholds and the method is described in Korkas and Fryzlewicz (2015) and Cho and Fryzlewicz (2012). See also the supplementary material for the former work. The function works for any sample size.

Usage

```
get.thres(n, q=.95, r=100, scales=NULL)
```

Arguments

`n`	The length of the time series.
`q`	The quantile of the r simulations.
`r`	Number of simulations.
`scales`	The wavelet periodogram scales to be used. If NULL (DEFAULT) then this is selected as described in the main text.

References

K. Korkas and P. Fryzlewicz (2015), Multiple change-point detection for non-stationary time series using Wild Binary Segmentation. (http://stats.lse.ac.uk/fryzlewicz/WBS_LSW/WBS_LSW.pdf)

K. Korkas and P. Fryzlewicz (2015), Supplementary material: Multiple change-point detection for non-stationary time series using Wild Binary Segmentation.

Cho, H. and Fryzlewicz, P. (2012). Multiscale and multilevel technique for consistent segmentation of nonstationary time series. Statistica Sinica, 22(1), 207-229.

`get.thres.ar` *Selection of thresholds by fitting an AR(p) model*

Description

The function returns data-driven thresholds and it is described in Korkas and Fryzlewicz (2015) where it is referred as Bsp1. See also the supplementary material for this work.

Usage

```
get.thres.ar(y, q=.95, r=100, scales=NULL)
```

Arguments

`y`	The time series.
`q`	The quantile of the r simulations.
`r`	Number of simulations.
`scales`	The wavelet periodogram scales to be used. If NULL (DEFAULT) then this is selected as described in the main text.

Author(s)

K. Korkas and P. Fryzlewicz

References

K. Korkas and P. Fryzlewicz (2015), Multiple change-point detection for non-stationary time series using Wild Binary Segmentation. (http://stats.lse.ac.uk/fryzlewicz/WBS_LSW/WBS_LSW.pdf)

K. Korkas and P. Fryzlewicz (2015), Supplementary material: Multiple change-point detection for non-stationary time series using Wild Binary Segmentation.

Examples

```
cps=seq(from=100,to=1200,by=350)
y=sim.pw.arma(N =1200,sd_u = c(1,1.5,1,1.5,1),
b.slope=rep(0.99,5),b.slope2 = rep(0.,5), mac = rep(0.,5),br.loc = cps)[[2]]
C_i=get.thres.ar(y=y, q=.95, r=100, scales=NULL)
wbs.lsw(y,M=1, C_i = C_i)$cp.aft
```

post.processing *Post-processing of the change-points*

Description

A function to control the number of change-points estimated from the WBS algorithm and to reduce the risk of over-segmentation.

Usage

```
post.processing(z,br,del=-1,epp=-1,C_i=NULL,scales=NULL)
```

Arguments

z	The wavelet periodogram matrix.
br	The change-points to be post-processed.
del	The minimum allowed size of a segment.
epp	A minimum adjustment for the bias present in $E^{(i)} - t, T$.
C_i	The CUSUM threshold.
scales	Which wavelet periodogram scales to be used.

References

K. Korkas and P. Fryzlewicz (2015), Multiple change-point detection for non-stationary time series using Wild Binary Segmentation. (http://stats.lse.ac.uk/fryzlewicz/WBS_LSW/WBS_LSW.pdf)

Examples

```
#### Generate a highly persistent time series with changing variance and of length 5,000
###Location of the change-points
cps=seq(from=1000,to=2800,by=200)
y=sim.pw.arma(N =3000,sd_u = c(1,1.5,1,1.5,1,1.5,1,1.5,1,1.5,1),
b.slope=rep(0.99,11),b.slope2 = rep(0.,11), mac = rep(0.,11),br.loc = cps)[[2]]
###Estimate the change points via Wild Binary Segmentation
beforeProcessing=wbs.lsw(y,M=0)$cp.bef
###Post-processing of the change points
post.processing(z=ews.trans(y,c(11,10,9,8)),br=beforeProcessing,C_i=tau.fun(y),scales=c(11,10,9,8))
```

sim.pw.ar — *Simulation of a piecewise constant AR(1) model*

Description

The function simulates a piecewise constant AR(1) model with multiple change-points

Usage

```
sim.pw.ar(N, sd_u, b.slope, br.loc)
```

Arguments

N	Length of the series.
sd_u	A vector of the innovation standard deviation for every segment.
b.slope	A vector of the AR(1) coefficients.
br.loc	A vector with the location of the change-points.

Value

A simulated series

Examples

```
cps=c(400,612)
y=sim.pw.ar(N =1024,sd_u = 1,b.slope=c(0.4,-0.6,0.5),br.loc=cps)[[2]]
ts.plot(y)
abline(v=cps,col="red")
```

sim.pw.ar2 — *Simulation of a piecewise constant AR(2) model*

Description

The function simulates a piecewise constant AR(2) model with multiple change-points

Usage

```
sim.pw.ar2(N, sd_u, b.slope, b.slope2, br.loc)
```

Arguments

N	Length of the series
sd_u	A vector of the innovation standard deviation for every segment
b.slope	A vector of the AR(1) coefficients
b.slope2	A vector of the AR(2) coefficients
br.loc	A vector with the location of the change-points

Value

A simulated series

Examples

```
cps=c(512,754)
y=sim.pw.ar2(N =1024,sd_u = 1,b.slope=c(0.9,1.68,1.32),
b.slope2 = c(0.0,-0.81,-0.81),br.loc = cps)[[2]]
ts.plot(y)
abline(v=cps,col="red")
```

---

sim.pw.arma *Simulation of a piecewise constant ARMA(p,q) model for p=2 and q=1*

Description

The function simulates a piecewise constant ARMA model with multiple change-points

Usage

```
sim.pw.arma(N, sd_u, b.slope, b.slope2, mac, br.loc)
```

Arguments

N	Length of the series
sd_u	A vector of the innovation standard deviation for every segment
b.slope	A vector of the AR(1) coefficients
b.slope2	A vector of the AR(2) coefficients
mac	A vector of the MA(1) coefficients
br.loc	A vector with the location of the change-points

Value

A simulated series

Examples

```
cps=c(125,532,704)
y=sim.pw.arma(N = 1024,sd_u = 1,b.slope=c(0.7,0.3,0.9,0.1),
b.slope2 = c(0,0,0,0), mac = c(0.6,0.3,0,-0.5),br.loc = cps)[[2]]
ts.plot(y)
abline(v=cps,col="red")
```

`tau.fun` *Universal thresholds*

Description

The function returns $C^{(i)}$. $C^{(i)}$ tends to increase as we move to coarser scales due to the increasing dependence in the wavelet periodogram sequences. Since the method applies to non-dyadic structures it is reasonable to propose a general rule that will apply in most cases. To accomplish this the $C^{(i)}$ are obtained for $T = 50, 100, ..., 6000$. Then, for each scale i the following regression is fitted

$C^{(i)} = c_0^{(i)} + c_1^{(i)}T + c_2^{(i)}\frac{1}{T} + c_3^{(i)}T^2 + \varepsilon.$

The adjusted R^2 was above 90% for all the scales. Having estimated the values for $\hat{c}_0^{(i)}, \hat{c}_1^{(i)}, \hat{c}_2^{(i)}, \hat{c}_3^{(i)}$ the values can be retrieved for any sample size T.

Usage

```
tau.fun(y)
```

Arguments

y	A time series

Value

Thresholds for every wavelet scale

Author(s)

K. Korkas and P. Fryzlewicz

References

P. Fryzlewicz (2014), Wild Binary Segmentation for multiple change-point detection. Annals of Statistics, 42, 2243-2281. (http://stats.lse.ac.uk/fryzlewicz/wbs/wbs.pdf)

K. Korkas and P. Fryzlewicz (2015), Multiple change-point detection for non-stationary time series using Wild Binary Segmentation. (http://personal.lse.ac.uk/KORKAS/papers.html)

Examples

```
cps=c(400,470)
set.seed(101)
y=sim.pw.ar(N =2000,sd_u = 1,b.slope=c(0.4,-0.6,0.5),br.loc=cps)[[2]]
#tau.fun(y) is the default value for C_i
#Binary segmentation
wbs.lsw(y,M=1)$cp.aft
#Wild binary segmentation
wbs.lsw(y,M=3500)$cp.aft
```

---

uh.wbs *The Wild Binary Segmentation algorithm*

Description

The function implements the Wild Binary Segmentation method and aggregates the change-points across the wavelet periodogram. Currently only the Method 2 of aggregation is implemented.

Usage

```
uh.wbs(z,C_i, del=-1, epp, scale,M=0,cstar=0.75)
```

Arguments

z	The wavelet periodogram matrix.
C_i	The CUSUM threshold.
del	The minimum allowed size of a segment.
epp	A minimum adjustment for the bias present in $E^{(i)} - t, T$.
scale	Which wavelet periodogram scales to be used.
M	The maximum number of random intervals drawn. If M=0 (DEFAULT) this is selected to be a linear function of the sample size of y. If M=1 then the segmentation is conducted via the Binary segmentation method.
cstar	This refers to the unbalanceness parameter $c_\star$.

Value

cp.bef	Returns the estimated change-points before post-processing
cp.aft	Returns the estimated change-points after post-processing

References

K. Korkas and P. Fryzlewicz (2015), Multiple change-point detection for non-stationary time series using Wild Binary Segmentation. (http://stats.lse.ac.uk/fryzlewicz/WBS_LSW/WBS_LSW.pdf)

Examples

```
#### Generate a highly persistent time series with changing variance and of length 5,000
###Location of the change-points
cps=seq(from=1000,to=2800,by=200)
y=sim.pw.arma(N =3000,sd_u = c(1,1.5,1,1.5,1,1.5,1,1.5,1,1.5,1),
b.slope=rep(0.99,11),b.slope2 = rep(0.,11), mac = rep(0.,11),br.loc = cps)[[2]]
###Estimate the change points via Binary Segmentation
wbs.lsw(y,M=1)$cp.aft
###Estimate the change points via Wild Binary Segmentation
wbs.lsw(y,M=0)$cp.aft
```

`wbs.lsw` *Change point detection for a nonstationary process using Wild Binary Segmentation*

Description

The function returns the estimated locations of the change-points in a nonstationary time series. Currently only the Method 2 of aggregation is implemented.

Usage

```
wbs.lsw(y, C_i = tau.fun(y), scales = NULL, M = 0, cstar = 0.75, lambda = 0.75)
```

Arguments

`y`	The time series.
`C_i`	A vector of threshold parameters for different scales.
`scales`	The wavelet periodogram scales to be used. If NULL (DEFAULT) then this is selected as described in the main text.
`M`	The maximum number of random intervals drawn. If M=0 (DEFAULT) this is selected to be a linear function of the sample size of y. If M=1 then the segmentation is conducted via the Binary segmentation method.
`cstar`	This refers to the unbalanceness parameter $c_\star$.
`lambda`	This parameter defines the maximum number of the wavelet periodogam scales. This is used if scales = NULL.

Value

`cp.bef`	Returns the estimated change-points before post-processing
`cp.aft`	Returns the estimated change-points after post-processing

Author(s)

K. Korkas and P. Fryzlewicz

References

K. Korkas and P. Fryzlewicz (2015), Multiple change-point detection for non-stationary time series using Wild Binary Segmentation. (http://stats.lse.ac.uk/fryzlewicz/WBS_LSW/WBS_LSW.pdf)

Examples

```
#### Generate a highly persistent time series with changing variance and of length 5,000
###Location of the change-points
cps=seq(from=1000,to=2800,by=200)
y=sim.pw.arma(N =3000,sd_u = c(1,1.5,1,1.5,1,1.5,1,1.5,1,1.5,1),
b.slope=rep(0.99,11),b.slope2 = rep(0.,11), mac = rep(0.,11),br.loc = cps)[[2]]
###Estimate the change points via Binary Segmentation
wbs.lsw(y,M=1)$cp.aft
###Estimate the change points via Wild Binary Segmentation
wbs.lsw(y,M=0)$cp.aft
```

Index