

Package ‘TED’

August 29, 2016

Type Package

Title Turbulence Time Series Event Detection and Classification

Version 1.1.1

Date 2014-10-02

Author Yanfei Kang, Danijel Belusic and Kate Smith-Miles

Maintainer Yanfei Kang <yanfei.kang@monash.edu>

Description TED performs Turbulence time series Event Detection and classification.

LazyLoad yes

Repository CRAN

Depends R (>= 3.0.2)

Imports foreach, zoo, fields, animation, geoR, tcltk, utils,
RcppArmadillo

Suggests doMC

NeedsCompilation no

License GPL (>= 2)

Date/Publication 2014-10-02 09:44:04

R topics documented:

ted-package	2
aniplotevents	3
CASES99	4
cbfs	5
cbfs_red	6
detrendc	7
eventCluster	8
eventDetection	10
eventExtraction	12
measures	13
noiseTests	14
plotevents	15
ts2mat	17
ur.za.fast	18

ted-package

Detect and classify events from turbulence time series

Description

TED performs Turbulence time series Event Detection and classification. The method consists of two steps. The event detection step locates and detects events by performing noise tests on sliding subsequences extracted from the time series. A subsequence is considered to be a potential event if its characteristics are significantly different from noise. The event is defined only if the consecutive sequence of potential events is long enough. This step does not rely on pre-assumption of events in terms of their magnitude, geometry, or stationarity. The main function `eventDetection` should be used for this step. The event classification step is to classify the events into groups with similar global characteristics. Each event is summarised using a feature vector, and then the events are clustered according to the Euclidean distances among the feature vectors. The main function `eventCluster` should be used for the classification step. Examples of event detection and classification can be found in the package for both artificial data and real world turbulence data.

Details

The package contains two main functions:

`eventDetection`: to detect events from time series as described in Kang et al. (2014b).

`eventCluster`: to classify the detect events from time series as described in Kang et al. (2014b).

The package also contains functions for visualising the events:

`plotevents`: to plot the detected and classified events.

`aniplotevents`: to generate a gif to visualise the event detection process.

Other sub-functions are:

`cbfs`: to generate an artificial event with white noise.

`cbfs_red`: to generate an artificial event with red noise.

`detrendc`: to conditionally detrend a time series.

`eventExtraction`: to extract events from the noise test results of a time series.

`measures`: to calculate statistical characteristics of an event.

`noiseTests`: to perform noise tests for a time series.

`ts2mat`: to reshape a vector into a matrix.

`ur.za.fast`: unit root test for events considering a structural break.

The real world turbulence dataset used in this package is available by loading:

`CASES99`: one day of 1-s averages of the thermocouple temperature data from CASES-99 dataset (Poulos et al. (2002)).

Author(s)

Yanfei Kang, Danijel Belusic and Kate Smith-Miles

Maintainer: Yanfei Kang <yanfei.kang@monash.edu>

References

- Yanfei Kang, Kate Smith-Miles, Danijel Belusic (2013). How to extract meaningful shapes from noisy time-series subsequences? *2013 IEEE Symposium on Computational Intelligence and Data Mining*, Singapore, 65-72. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6597219&isnumber=6597208>.
- Yanfei Kang, Danijel Belusic, Kate Smith-Miles (2014a). Detecting and Classifying Events in Noisy Time Series. *J. Atmos. Sci.*, **71**, 1090-1104. <http://dx.doi.org/10.1175/JAS-D-13-0182.1>.
- Yanfei Kang, Danijel Belusic, Kate Smith-Miles (2014b). Classes of structures in the stable atmospheric boundary layer. Submitted to Quarterly Journal of the Royal Meteorological Society.
- Xiaozhe Wang, Kate Smith-Miles and Rob Hyndman (2005). Characteristic-Based Clustering for Time Series Data. *Data Mining and Knowledge Discovery*. **13**(3), 335-364. <http://dx.doi.org/10.1007/s10618-005-0039-x>.
- Gregory S. Poulos, William Blumen, David C. Fritts, Julie K. Lundquist, Jielun Sun, Sean P. Burns, Carmen Nappo, Robert Banta, Rob Newsom, Joan Cuxart, Enric Terradellas, Ben Balsley, and Michael Jensen. CASES-99: A comprehensive investigation of the stable nocturnal boundary layer (2002). *Bulletin of the American Meteorological Society*, **83**(4):555-581.

 aniplothevents

 Generate a gif to visualise the event detection process

Description

This function generates a gif file demonstrating how the event detection process is implemented.

Usage

```
aniplothevents(x, w, noiseType = c("white", "red"), alpha = 0.05,
  main = "Animation plot of events", xlab = "t", ylab = "x",
  movie.name = "animation.gif", interval = 0.05, ani.width = 1000,
  ani.height = 400, outdir = getwd())
```

Arguments

x	a vector or a time series.
w	a scalar specifying the size of the sliding window.
noiseType	background noise type assumed for x. There are two options: white noise or red noise.
alpha	the significance level. When the noise test p value of the subsequence is smaller than this significance level, it is defined as a potential event.
main	title of the animation plot; default is 'Animation plot of event detection'.
xlab	x label of the animation plot; default is 't'.
ylab	y label of the animation plot; default is 'x'.

<code>movie.name</code>	name of the output gif file; default is 'animation.gif'.
<code>interval</code>	a positive number to set the time interval of the animation (unit in seconds); default is 0.05.
<code>ani.width</code>	width of the gif file (unit in px), default is 1000.
<code>ani.height</code>	height of the gif file (unit in px); default is 400.
<code>outdir</code>	character: specify the output directory when exporting the animations; default to be the current working directory.

Value

...

References

Yihui Xie (2013). Animation: An R Package for Creating Animations and Demonstrating Statistical Methods. *Journal of Statistical Software*, **53**(1), 1-27. <http://www.jstatsoft.org/v53/i01/>.

See Also

[noiseTests](#), [eventExtraction](#), [plotevents](#)

Examples

```
set.seed(123)
# generate an artificial time series
x=c(rnorm(128),cbfs(type='box'),rnorm(128),cbfs(type='rc'),rnorm(128))
# generate a gif file to show the event detection process
## Not run:
aniplotevents(x,w=128,noiseType='white',outdir=getwd())

## End(Not run)
```

CASES99

One day of 1-s averages of the thermocouple temperature data from CASES-99 dataset

Description

These are 1-s averages of the CASES-99 (Poulos et al. 2002) thermocouple temperature data at the seventh level (9.5 m) from 1100 LST 5 October to 1100 LST 6 October.

Usage

```
data(CASES99)
```

Details

Cooperative Atmosphere-Surface Exchange Study (CASES-99) was conducted over a relatively flat-terrain rural grassland site near Leon, Kansas, during October 1999. As a part of the extensive observations, a 60-m tower was equipped with thermocouples at 34 vertical levels (0.23, 0.63, 2.3 m, and every 1.8 m above 2.3 m) that sampled air temperature five times per second (Sun et al. 2012), while 20-Hz sonic anemometer measurements were taken at seven levels (1.5, 5, 10, 20, 30, 40, 50, and 55 m). 1-s averages of the CASES-99 thermocouple temperature data at the seventh level (9.5 m) from 1100 LST 5 October to 1100 LST 6 October are taken as an example for detection and clustering of events.

Source

Gregory S. Poulos, William Blumen, David C. Fritts, Julie K. Lundquist, Jielun Sun, Sean P. Burns, Carmen Nappo, Robert Banta, Rob Newsom, Joan Cuxart, Enric Terradellas, Ben Balsley, and Michael Jensen. CASES-99: A comprehensive investigation of the stable nocturnal boundary layer (2002). *Bulletin of the American Meteorological Society*, **83**(4):555-581.

Examples

```
data(CASES99)
```

```
cbfs
```

```
Generate an artificial event with white noise
```

Description

This function generates a box, cliff-ramp, ramp-cliff or a sine function with different levels of white noise as the background noise. Length of the generated event is 128. Generation of events are similar to that of Cylinder-Bell-Funnel dataset in the reference below (Keogh and Lin 2005).

Usage

```
cbfs(type = c("box", "rc", "cr", "sine"), A = 10, sigma = 1)
```

Arguments

type	type of the event to be generated. There are four options: 'box', 'rc', 'cr', 'sine' representing a box, cliff-ramp, ramp-cliff or a sine function.
A	amplitude of the event; default is 10.
sigma	a scalar specifying the level of white noise. Default is 1, which means the standard deviation of noise is 1.

Value

an artificial event with white noise.

References

Eamonn Keogh and Jessica Lin (2005). Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowl. Inf. Syst.*, **8**(2), 154-177. <http://dblp.uni-trier.de/db/journals/kais/kais8.html#KeoghL05>.

Yanfei Kang, Kate Smith-Miles, Danijel Belusic (2013). How to extract meaningful shapes from noisy time-series subsequences? *2013 IEEE Symposium on Computational Intelligence and Data Mining*, Singapore, 65-72. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6597219&isnumber=6597208>.

Yanfei Kang, Danijel Belusic, Kate Smith-Miles (2014). Detecting and Classifying Events in Noisy Time Series. *J. Atmos. Sci.*, **71**, 1090-1104. <http://dx.doi.org/10.1175/JAS-D-13-0182.1>.

Examples

```
# generate a box function with white noise
set.seed(123)
x1 = cbfs(type = 'box', sigma = 1)
# generate a box function with higher level noise
set.seed(123)
x2 = cbfs(type = 'box', sigma = 3)
# plot them
par(mfrow=c(1,2))
plot(x1,type='l',xlab='t',ylab=expression(x[1]))
plot(x2,type='l',xlab='t',ylab=expression(x[2]))
```

cbfs_red

Generate an artificial event with red noise

Description

This function generates a box, cliff-ramp, ramp-cliff or a sine function with red noise (AR(1)) as the background noise. Length of the generated event is 128.

Usage

```
cbfs_red(type = c("box", "rc", "cr", "sine"), A = 10, s = 1,
  coeff = 0.5)
```

Arguments

type	type of the event to be generated. There are four options: "box", "rc", "cr", "sine" representing a box, cliff-ramp, ramp-cliff or a sine function.
A	amplitude of the event; default is 10.
s	standard deviation of the AR(1) model innovations. Default is 1.
coeff	coefficient of the AR(1) process, which is used to control the level of red noise. Default is 0.5.

Value

an artificial event with red noise.

Examples

```
# generate a box function with red noise
set.seed(123)
x = cbfs_red(type = 'box', coeff=0.5, s=1, A=10)
# plot it
plot(x,type='l',xlab='t',ylab='x')
```

detrendc

Conditionally detrend a time series

Description

This function detrends a time series when its linear trend is more significant than a threshold.

Usage

```
detrendc(x, thres = 0.85)
```

Arguments

x a vector or time series.

thres a scalar specifying the threshold. When the adjusted R square coefficient of the linear fitting is larger than this threshold, the linear trend is subtracted from the original time series. Default is 0.85.

Value

detrended x.

Examples

```
t=seq(0.001,1,0.001)
set.seed(123)
x=10*t+rnorm(1000)
dtrx=detrendc(x)
# plot the simulated x
plot(t,x,ty='l',xlab='t',ylab='x')
# plot the detrended x
lines(t,dtrx,col=2)
legend(0,12,legend=c('x', 'detrended x'),col=c(1,2),lty=1)
```

eventCluster	<i>Cluster detected events</i>
--------------	--------------------------------

Description

This function groups the detected events into clusters.

Usage

```
eventCluster(events, k0)
```

Arguments

events	an object of class 'events'.
k0	the number of clusters.

Details

The clustering is based on statistical characteristics of event. Each extracted event is first described using a feature vector, and then the events are clustered according to the Euclidean distances among the feature vectors. Note that before clustering, we apply principal component analysis (PCA) to the feature vectors to reduce the correlation as well as the dimension of the feature space.

Value

a list consisting of:

cl	a vector indicating which cluster each event belongs to.
center	a matrix which gives cluster centroids.
pca	PCA results for characteristics of the detected events.

References

- Xiaozhe Wang, Kate Smith-Miles and Rob Hyndman (2005). Characteristic-Based Clustering for Time Series Data. *Data Mining and Knowledge Discovery*. **13**(3), 335-364. <http://dx.doi.org/10.1007/s10618-005-0039-x>
- Yanfei Kang, Danijel Belusic, Kate Smith-Miles (2014). Detecting and Classifying Events in Noisy Time Series. *J. Atmos. Sci.*, **71**, 1090-1104. <http://dx.doi.org/10.1175/JAS-D-13-0182.1>.
- Gregory S. Poulos, William Blumen, David C. Fritts, Julie K. Lundquist, Jielun Sun, Sean P. Burns, Carmen Nappo, Robert Banta, Rob Newsom, Joan Cuxart, Enric Terradellas, Ben Balsley, and Michael Jensen. CASES-99: A comprehensive investigation of the stable nocturnal boundary layer (2002). *Bulletin of the American Meteorological Society*, **83**(4):555-581.

See Also

[measures](#)

Examples

```
#####
# An artificial example
#####
set.seed(123)
n <- 128
types <- c('box','rc','cr','sine')
shapes <- matrix(NA,20,n)
for (i in 1:20){
  shapes[i,] <- cbfs(type=types[sample(1:4,1)])
}
whitenoise <- ts2mat(rnorm(128*20),128)
# generate x which randomly combine the four types of events with each two of them
# separated by noise
x <- c(rnorm(128),t(cbind(shapes,whitenoise)))
## Not run:
plot(x,ty='l')

## End(Not run)
# specify a sliding window size
w <- 128
# specify a significant level
alpha <- 0.05
# event detection
## Not run:
events <- eventDetection(x,w,'white',parallel=FALSE,alpha, 'art')
# clustering
cc <- eventCluster(events,4)
myclkm <- cc$c1

## End(Not run)
#####
# CASES-99 dataset (9.5m)
#####
# a sliding window length chosen by the user
w <- 120
# specify a significant level
alpha <- 0.05
data(CASES99)
## Not run:
CASESevents <- eventDetection(CASES99,w,'red',parallel=FALSE,0.05,'real')
cc <- eventCluster(CASESevents,3)
cc$center
myclkm <- cc$c1
# visualise the clustering in 2-dimension PCA space
pc.cr <- cc$pca
pca.dim1 <- pc.cr$scores[,1]
pca.dim2 <- pc.cr$scores[,2]
plot(pca.dim1,pca.dim2,col=myclkm+1,main='PCA plots for k-means clustering',pch=16)

## End(Not run)
```

eventDetection *Detect events from time series*

Description

This function finds events from a time series.

Usage

```
eventDetection(x, w, noiseType = c("white", "red"), parallel = FALSE,
              alpha = 0.05, data = c("art", "real"))
```

Arguments

x	a vector or time series.
w	size of the sliding window.
noiseType	background noise type assumed for x. There are two options: white noise or red noise.
parallel	logical, if TRUE then codes are executed in parallel using foreach package. The user must register a parallel backend to use by the doMC package.
alpha	the significance level. When the noise test p value of the subsequence is smaller than this significance level, it is defined as a potential event. Default is 0.05.
data	type of data being analysed. There are two options: 'art' if analysed data is artificial data and 'real' if analysed data is real world turbulence data. Please see the details in Kang et al. (2014).

Value

an object of class 'events' with the components listed below:

x	the original time series.
start	a vector consisting of starting points of events.
end	a vector consisting of ending points of events.
nevents	number of detected events.

References

Yanfei Kang, Danijel Belusic, Kate Smith-Miles (2014): Detecting and Classifying Events in Noisy Time Series. *J. Atmos. Sci.*, **71**, 1090-1104. <http://dx.doi.org/10.1175/JAS-D-13-0182.1>.

Gregory S. Poulos, William Blumen, David C. Fritts, Julie K. Lundquist, Jielun Sun, Sean P. Burns, Carmen Nappo, Robert Banta, Rob Newsom, Joan Cuxart, Enric Terradellas, Ben Balsley, and Michael Jensen. CASES-99: A comprehensive investigation of the stable nocturnal boundary layer (2002). *Bulletin of the American Meteorological Society*, **83**(4):555-581.

See Also

[noiseTests](#), [eventExtraction](#), [plotevents](#)

Examples

```
#####
# 1st art eg (white noise)
#####
set.seed(123)
n=128
types=c('box','rc','cr','sine')
shapes=matrix(NA,20,n)
for (i in 1:20){
  shapes[i,]=cbfs(type=types[sample(1:4,1)])
}
whitenoise=ts2mat(rnorm(128*20),128)
# generate x which randomly combine the four types of events with each two of them
# separated by noise
x=c(rnorm(128),t(cbind(shapes,whitenoise)))
plot(x,ty='l')
# specify a sliding window size and significant level
## Not run:
w=128; alpha=0.05
events=eventDetection(x,w,'white',parallel=FALSE,alpha,'art')

## End(Not run)
#####
# 2nd art eg (red noise)
#####
set.seed(123)
# set a red noise level
coeff=0.5;s=1
# generated x with red noise as the background; this time series is the one used in
# Kang et al. (2014)
x=c(arima.sim(list(order = c(1,0,0),ar=coeff),n=500,sd=s),
     cbfs_red('rc'),arima.sim(list(order = c(1,0,0),ar=coeff),n=400,sd=s),
     cbfs_red('cr'),arima.sim(list(order = c(1,0,0),ar=coeff),n=400,sd=s),
     cbfs_red('box'),arima.sim(list(order = c(1,0,0),ar=coeff),n=400,sd=s),
     cbfs_red('sine'),arima.sim(list(order = c(1,0,0),ar=coeff),n=1000,sd=s),
     arima.sim(list(order = c(1,0,0),ar=0.8),n=1100,sd=4))
# specify a sliding window size and significant level
## Not run:
w=128; alpha=0.05
# event detection
events=eventDetection(x,w,'red',parallel=FALSE,alpha,'art')

## End(Not run)
#####
# CASES-99 dataset (9.5m)
#####
# window size which needs to be chosen by the user
w=120
```

```
# specify a significant level
alpha=0.05
# event detection from CASES99 data
## Not run:
data(CASES99)
CASESevents=eventDetection(CASES99,w,'red',parallel=FALSE,alpha,'real')

## End(Not run)
```

eventExtraction *Extract events from time series*

Description

This function returns the starting and ending points of events according to the noise test results from a time series.

Usage

```
eventExtraction(tests, w, alpha = 0.05)
```

Arguments

tests	test p values from the noist tests for the subsequences.
w	sliding window size.
alpha	the significance level. When the noise test p value of the subsequence is smaller than this significance level, it is a potential event. Default is 0.05.

Value

a list consisting:

start	a vector consisting of starting points of events.
end	a vector consisting of ending points of events.
tests	smoothed test p value series.
nevents	number of detected events.

References

Yanfei Kang, Danijel Belusic, Kate Smith-Miles (2014): Detecting and Classifying Events in Noisy Time Series. *J. Atmos. Sci.*, **71**, 1090-1104. <http://dx.doi.org/10.1175/JAS-D-13-0182.1>.

`measures`*Calculate statistical characteristics of an event*

Description

This function calculates statistical characteristics for detected events.

Usage

```
measures(x)
```

Arguments

`x` a time series

Details

Measures used here are standard deviation, kurtosis, skewness, HD (the absolute Difference between averages of the first and second Half), nonsmoothness, test statistic of PP test and ZA test, and maximum, minimum, and kurtosis of the first-order difference of the events. Please see the reference for details (Kang et al. 2014).

Value

a vector consisting of statistical characteristics of event `x`

References

Yanfei Kang, Danijel Belusic, Kate Smith-Miles (2014). Classes of structures in the stable atmospheric boundary layer. Submitted to Quarterly Journal of the Royal Meteorological Society.

See Also

[eventCluster](#)

Examples

```
set.seed(123)
n=128
measures(cbfs('box'))
measures(cbfs('sine'))
```

noiseTests	<i>Perform noise tests for a time series</i>
------------	--

Description

This function performs noise tests on the sliding subsequences extracted from a time series.

Usage

```
noiseTests(x, w, noiseType = c("white", "red"), parallel = FALSE)
```

Arguments

x	a vector or a time series.
w	a scalar specifying the size of the sliding window.
noiseType	background noise assumed for x. There are two options: “white” or “red” .
parallel	logical, if TRUE then codes are executed in parallel using the foreach package. The user must register a parallel backend to use by the doMC package.

Details

When using this function, the user needs to choose the background noise type via noiseType according to the application context. In atmospheric turbulence, red noise is used. We first use the Phillips-Perron (PP) Unit Root Test to test for the unit root process. For the stationary processes, red noise tests are performed to test for events. For those cases tested to be unit root processes, we have to take into consideration a special situation when there is a structural break in the process. The reason comes from the difficulty for PP test to distinguish random walk processes from a stationary process contaminated by a structural break, both of which result in non-rejection of the null hypothesis. Random-walk processes are not considered as events since they are known to be brownian noise, but stationary processes with structure breaks are, so it is essential to distinguish them. To this end, an additional test called Zivot & Andrews (ZA) unit root test is introduced. This test allows for a structural break in either the intercept or in the slope of the trend function of the underlying series. Rejection of the null hypothesis indicates a potential event (stationary process with a structural break). Random walk processes result in non-rejection of the null hypothesis.

Value

test p value series for the time series x.

References

- Pierre Perron (1998). Trends and random walks in macroeconomic time series: Further evidence from a new approach. *Journal of economic dynamics and control*, **12**(2), 297-332. [http://dx.doi.org/10.1016/0304-3932\(82\)90012-5](http://dx.doi.org/10.1016/0304-3932(82)90012-5).
- Eric Zivot and Donald W K Andrews (1992). Further evidence on the great crash, the oil-price shock, and the unit-root hypothesis. *Journal of Business & Economic Statistics*, **20**(1), 25-44. <http://dx.doi.org/10.1198/073500102753410372>.

Yanfei Kang, Danijel Belusic and Kate Smith-Miles (2014). Detecting and Classifying Events in Noisy Time Series. *J. Atmos. Sci.*, **71**, 1090-1104. <http://dx.doi.org/10.1175/JAS-D-13-0182.1>.

See Also

[eventExtraction](#), [plotevents](#)

Examples

```
set.seed(123)
n=128
types=c('box','rc','cr','sine')
shapes=matrix(NA,20,n)
for (i in 1:20){
  shapes[i,]=cbfs(type=types[sample(1:4,1)])
}
whitenoise=ts2mat(rnorm(128*20),128)
# generate x which randomly combine the four types of events with each two of them
# separated by noise
x=c(t(cbind(shapes,whitenoise)))
plot(x,ty='l')
w=128
# execute loops sequentially
## Not run:
tests=noiseTests(x,w,'white',parallel=FALSE)

## End(Not run)
# execute loops in parallel using doMC package (for non-Windows users)
## Not run:
tests=noiseTests(x,w,'white',parallel=TRUE)

## End(Not run)
```

plotevents

Plot the detected events

Description

This function plots the detected events from a time series.

Usage

```
plotevents(events, cluster = FALSE, mycl, ...)
```

Arguments

events	an object of class 'events'.
cluster	logical, if TRUE then the detected events are highlighted using different colors for different clusters
mycl	a vector specifying which cluster each event belongs to
...	other arguments that can be passed to plot

Value

...

References

Yanfei Kang, Danijel Belusic and Kate Smith-Miles (2014). Detecting and Classifying Events in Noisy Time Series. *J. Atmos. Sci.*, **71**, 1090-1104. <http://dx.doi.org/10.1175/JAS-D-13-0182.1>.

See Also

[noiseTests](#), [eventExtraction](#), [eventDetection](#)

Examples

```
#####
# 1st art eg (white noise)
#####
set.seed(123)
n=128
types=c('box','rc','cr','sine')
shapes=matrix(NA,20,n)
for (i in 1:20){
  shapes[i,]=cbfs(type=types[sample(1:4,1)])
}
whitenoise=ts2mat(rnorm(128*20),128)
# generate x which randomly combine the four types of events with each two of them
# separated by noise
x=c(rnorm(128),t(cbind(shapes,whitenoise)))
plot(x,ty='l')
w=128; alpha=0.05
# event detection
## Not run:
events=eventDetection(x,w,'white',FALSE,alpha,'art')
clustering events
cc=eventCluster(events,4)
myclkm=cc$cl
# plot the clustered events
plotevents(events,cluster=TRUE, myclkm)

## End(Not run)
#####
```



```

# 2nd art eg (red noise)
#####
set.seed(123)
# generate a time series with red noise; this is the same with the one used
# in Kang et al. (2014)
coeff=0.5;s=1
x=c(arima.sim(list(order = c(1,0,0),ar=coeff),n=500,sd=s),
     cbfs_red('rc'),arima.sim(list(order = c(1,0,0),ar=coeff),n=400,sd=s),
     cbfs_red('cr'),arima.sim(list(order = c(1,0,0),ar=coeff),n=400,sd=s),
     cbfs_red('box'),arima.sim(list(order = c(1,0,0),ar=coeff),n=400,sd=s),
     cbfs_red('sine'),arima.sim(list(order = c(1,0,0),ar=coeff),n=1000,sd=s),
     arima.sim(list(order = c(1,0,0),ar=0.8),n=1100,sd=4))
w=128; alpha=0.05
# event detection
## Not run:
events=eventDetection(x,w,'red',parallel=FALSE,alpha,'art')
# plot events without clustering
plotevents(events)

## End(Not run)

```

ts2mat

Reshape a vector into a matrix

Description

This function reshapes a vector into a matrix whose row elements are taken from the vector. Orders of elements keep unchanged from the vector.

Usage

```
ts2mat(x, w)
```

Arguments

x	a vector or a time series
w	a number specifying number of columns of the matrix

Value

a matrix

Examples

```

x=ts2mat(c(1:(128*20)),128)
dim(x)
x[1,1:20]

```

`ur.za.fast`*Unit root test for events considering a structural break*

Description

This function performs the Zivot & Andrews unit root test, which allows a break at an unknown point in either the intercept, the linear trend or in both.

Usage

```
ur.za.fast(y, model = c("intercept", "trend", "both"), lag = NULL)
```

Arguments

<code>y</code>	a vector or a time series.
<code>model</code>	Three choices: "intercept", "trend" or "both".
<code>lag</code>	a scalar chosen as lag.

Details

This function is written referring to the `ur.za` function in the **urza** package (Pfaff 2008), but it speeds up execution using the **ReppArmadillo** package. Allowing a structural break, this function returns `flag` to be 0 if the time series is stationary and 1 if it is a unit root process.

Value

a list consisting of:

<code>flag</code>	0 if the time series is stationary; 1 if it is a unit root process.
<code>teststat</code>	ZA unit root test statistic.

References

Eric Zivot and Donald W K Andrews (1992). Further evidence on the great crash, the oil-price shock, and the unit-root hypothesis. *Journal of Business & Economic Statistics*, **20**(1), 25-44. <http://dx.doi.org/10.1198/073500102753410372>.

Pfaff, Bernhard (2008). *Analysis of Integrated and Cointegrated Time Series with R*. Second Edition. Springer, New York. <http://www.springer.com/statistics/statistical+theory+and+methods/book/978-0-387-75966-1>.

See Also

[noiseTests](#)

Examples

```
# this is a box function
set.seed(123)
x=cbfs_red('box')
ur.za.fast(x, 'both')
# this is a cliff-ramp
set.seed(123)
x=cbfs_red('cr')
ur.za.fast(x, 'both')
# this is a random walk process
set.seed(123)
x=cumsum(rnorm(300))
ur.za.fast(x, 'both')
```

Index

*Topic **datasets**

CASES99, [4](#)

[aniplotevents](#), [2](#), [3](#)

CASES99, [2](#), [4](#)

[cbfs](#), [2](#), [5](#)

[cbfs_red](#), [2](#), [6](#)

[detrndc](#), [2](#), [7](#)

[eventCluster](#), [2](#), [8](#), [13](#)

[eventDetection](#), [2](#), [10](#), [16](#)

[eventExtraction](#), [2](#), [4](#), [11](#), [12](#), [15](#), [16](#)

[measures](#), [2](#), [8](#), [13](#)

[noiseTests](#), [2](#), [4](#), [11](#), [14](#), [16](#), [18](#)

[plotevents](#), [2](#), [4](#), [11](#), [15](#), [15](#)

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

[ts2mat](#), [2](#), [17](#)

[ur.za.fast](#), [2](#), [18](#)