Package 'funtimes'

February 4, 2018

Type Package

Title Functions for Time Series Analysis

Version 5.0

Date 2018-02-04

Depends R (>= 3.0.0)

Imports dbscan, Jmisc, Kendall

License GPL (>= 2)

Description Includes non-parametric estimators and tests for time series analysis. The functions are to test for presence of possibly non-monotonic trends and for synchronism of trends in multiple time series, using modern bootstrap techniques and robust non-parametric difference-based estimators.

NeedsCompilation no

Author Vyacheslav Lyubchich [aut, cre],
Yulia R. Gel [aut],
Calvin Chu [ctb],
Xin Huang [ctb],
Ethan D. Schaeffer [ctb],
Xingyu Wang [ctb]

Maintainer Vyacheslav Lyubchich <lyubchic@umces.edu>

Repository CRAN

Date/Publication 2018-02-04 21:14:47 UTC

R topics documented:

`ARest` *Estimation of Autoregressive (AR) Parameters*

Description

Estimate parameters ϕ of autoregressive time series model

$$X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + e_t,$$

by default using robust difference-based estimator and Bayesian information criterion (BIC) to select the order p. This function is mainly employed for time series filtering in functions `sync.test` and `wavk.test`.

Usage

```
ARest(x, ar.order = NULL, ar.method = "HVK", BIC = TRUE)
```

Arguments

`x`	a time series vector.
`ar.order`	order of autoregressive model when `BIC = FALSE`, or the maximal order for BIC-based filtering. Default is `round(10*log10(length(x)))`.
`ar.method`	method of estimating autoregression coefficients. Default `"HVK"` delivers robust difference-based estimates by Hall and Van Keilegom (2003). Alternatively, options of `ar` function can be used, such as `"burg"`, `"ols"`, `"mle"`, and `"yw"`.
`BIC`	logical value indicates whether the order of autoregressive filter should be selected by Bayesian information criterion (BIC). If `TRUE` (default), models of orders p = 0,1,...,`ar.order` or p = 0,1,...,`round(10*log10(length(x)))` are considered, depending on whether `ar.order` is defined or not.

Details

The same formula for BIC is used consistently for all methods:

$$BIC = n \ln(\hat{\sigma}^2) + k \ln(n),$$

where n = `length(x)`, $k = p + 1$.

Value

A vector of estimated AR coefficients. Returns `numeric(0)` if the final $p = 0$.

Author(s)

Vyacheslav Lyubchich

References

Hall, P. and Van Keilegom, I. (2003). Using difference-based methods for inference in nonparametric regression with time series errors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65: 443–456. DOI: 10.1111/1467-9868.00395

See Also

`ar`, `HVK`, `sync.test`, `wavk.test`

Examples

```
# Fix seed for reproducible simulations:
set.seed(1)

#Simulate some time series, possibly with trend:
n <- 100
Y <- arima.sim(n = n, list(order = c(2, 0, 0), ar = c(-0.7, -0.1)))
plot.ts(Y)

#Estimate the coefficients:
ARest(Y) #HVK by default
ARest(Y, ar.method = "yw") #Yule--Walker
ARest(Y, ar.method = "burg") #Burg
```

BICC *BIC-Based Spatio-Temporal Clustering*

Description

Apply the algorithm of unsupervised spatio-temporal clustering TRUST (Ciampi et al., 2010), with automatic selection of its tuning parameters `Delta` and `Epsilon` based on Bayesian information criterion (BIC).

Usage

```
BICC(X, Alpha = NULL, Beta = NULL, Theta = 0.8, p, w, s)
```

Arguments

X	a matrix of time series to be clustered (time series in columns).
Alpha	lower limit of the time series domain, passed to CSlideCluster.
Beta	upper limit of the time series domain, passed to CSlideCluster.
Theta	connectivity parameter, passed to CSlideCluster.
p	number of layers (time series observations) in each slide.
w	number of slides in each window.
s	step to shift a window, calculated in number of slides. Recommended values are 1 (overlapping windows) or equal to w (non-overlapping windows).

Details

This is the upper-level function for time series clustering. It exploits the functions CWindowCluster and CSlideCluster to cluster time series based on closeness and homogeneity measures. Clustering is performed multiple times with a range of equidistant values for the parameters Delta and Epsilon, then optimal parameters Delta and Epsilon along with the corresponding clustering results are shown (see Schaeffer et al., 2016, for more details).

The total length of time series (number of levels, i.e., nrow(X)) should be divisible by p.

Value

A list with the following elements:

Delta.optimal	'optimal' value for the clustering parameter Delta.
Epsilon.optimal	'optimal' value for the clustering parameter Epsilon.
Clusters	vector of length ncol(X) with cluster labels.
IC	values of the information criterion (BIC) for each considered combination of Delta (rows) and Epsilon (columns).
Delta.all	vector of considered values for Delta.
Epsilon.all	vector of considered values for Epsilon.

Author(s)

Ethan Schaeffer, Vyacheslav Lyubchich

References

Ciampi, A., Appice, A., and Malerba, D. (2010). Discovering trend-based clusters in spatially distributed data streams. In *International Workshop of Mining Ubiquitous and Social Environments*, pages 107–122.

Schaeffer, E. D., Testa, J. M., Gel, Y. R., and Lyubchich, V. (2016). On information criteria for dynamic spatio-temporal clustering. In A. Banerjee et al. (Eds.) *Proceedings of the 6th International Workshop on Climate Informatics: CI 2016*. NCAR Technical Note NCAR/TN-529+PROC, September 2016, pages 5–8. DOI: 10.5065/D6K072N6

See Also

CSlideCluster, CWindowCluster, purity

Examples

```
# Fix seed for reproducible simulations:
set.seed(1)

##### Example 1
# Similar to Schaeffer et al. (2016), simulate 3 years of monthly data
#for 10 locations and apply clustering:
# 1.1 Simulation
T <- 36 #total months
N <- 10 #locations
phi <- c(0.5) #parameter of autoregression
burn <- 300 #burn-in period for simulations
X <- sapply(1:N, function(x)
            arima.sim(n = T + burn,
            list(order = c(length(phi), 0, 0), ar = phi)))[(burn + 1):(T + burn),]
colnames(X) <- paste("TS", c(1:dim(X)[2]), sep = "")

# 1.2 Clustering
# Assume that information arrives in year-long slides or data chunks
p <- 12 #number of time layers (months) in a slide
# Let the upper level of clustering (window) be the whole period of 3 years, so
w <- 3 #number of slides in a window
s <- w #step to shift a window, but it does not matter much here as we have only one window of data
tmp <- BICC(X, p = p, w = w, s = s)

# 1.3 Evaluate clustering
# In these simulations, it is known that all time series belong to one class,
#since they were all simulated the same way:
classes <- rep(1, 10)
# Use the information on the classes to calculate clustering purity:
purity(classes, tmp$Clusters[1,])

##### Example 2
# 2.1 Modify time series and update classes accordingly:
# Add a mean shift to a half of the time series:
X2 <- X
X2[, 1:(N/2)] <- X2[, 1:(N/2)] + 3
classes2 <- rep(c(1, 2), each = N/2)

# 2.2 Re-apply clustering procedure and evaluate clustering purity:
tmp2 <- BICC(X2, p = p, w = w, s = s)
tmp2$Clusters
purity(classes2, tmp2$Clusters[1,])
```

CExpandSlideCluster *Slide-Level Time Series Cluster Expansion*

Description

This is an auxiliary function to expand a slide-level time series cluster, based on Ciampi et al. (2010).

Usage

```
CExpandSlideCluster(u, Xuncl, Alpha, Beta, Delta, Theta)
```

Arguments

u	a time series vector — a seed to expand the cluster.
Xuncl	a time series vector (of the same length as u) or a matrix (time series in columns) containing unclustered time series.
Alpha	lower limit of the time series domain.
Beta	upper limit of the time series domain.
Delta	closeness parameter, a real value in [0,1].
Theta	connectivity parameter, a real value in [0,1].

Value

A vector of logical values indicating which time series in Xuncl should be included in the slide-level cluster with u.

Author(s)

Vyacheslav Lyubchich

References

Ciampi, A., Appice, A., and Malerba, D. (2010). Discovering trend-based clusters in spatially distributed data streams. In *International Workshop of Mining Ubiquitous and Social Environments*, pages 107–122.

See Also

CNeighbor, CHomogeneity, CSlideCluster, CExpandWindowCluster, CWindowCluster

Examples

```
set.seed(123)
u <- rnorm(10)
Xuncl <- matrix(rt(50, 5), 10, 5)
Alpha <- min(cbind(u, Xuncl))
Beta <- max(cbind(u, Xuncl))
CExpandSlideCluster(u, Xuncl, Alpha, Beta, Delta = 0.15, Theta = 0.8)
```

CExpandWindowCluster *Window-Level Time Series Cluster Expansion*

Description

This is an auxiliary function to expand a window-level time series cluster, based on Ciampi et al. (2010).

Usage

```
CExpandWindowCluster(e, Euncl)
```

Arguments

e	a vector of logical values identifying which time series among Euncl were clustered together with e over at least w*Epsilon slides within a window (see Definition 7 by Ciampi et al., 2010). This is a seed for window-level clustering.
Euncl	a square matrix identifying the binary window cluster relation for yet unclustered time series.

Value

A vector of logical values indicating which time series in Euncl should be included in the window-level cluster with e.

Author(s)

Vyacheslav Lyubchich

References

Ciampi, A., Appice, A., and Malerba, D. (2010). Discovering trend-based clusters in spatially distributed data streams. In *International Workshop of Mining Ubiquitous and Social Environments*, pages 107–122.

See Also

CNeighbor, CHomogeneity, CSlideCluster, CExpandSlideCluster, CWindowCluster

Examples

```
set.seed(123)
e <- sample(c(TRUE, FALSE), 5, replace = TRUE)
Euncl <- matrix(sample(c(TRUE, FALSE), 5, replace = TRUE), 5, 5)
CExpandWindowCluster(e, Euncl)
```

CHomogeneity *Time Series Cluster Homogeneity*

Description

This is an auxiliary function to check homogeneity of time series cluster, based on Definition 4 by Ciampi et al. (2010).

Usage

```
CHomogeneity(Bu, Bv, Alpha, Beta, Delta)
```

Arguments

Bu	bucket of time series already included in the cluster.
Bv	bucket of time series (neighbors) for potential inclusion in the cluster.
Alpha	lower limit of the time series domain.
Beta	upper limit of the time series domain.
Delta	closeness parameter, a real value in [0,1].

Value

A logical value indicating whether time series in Bu and Bv form a homogeneous cluster.

Author(s)

Vyacheslav Lyubchich

References

Ciampi, A., Appice, A., and Malerba, D. (2010). Discovering trend-based clusters in spatially distributed data streams. In *International Workshop of Mining Ubiquitous and Social Environments*, pages 107–122.

See Also

CNeighbor, CExpandSlideCluster, CSlideCluster, CExpandWindowCluster, CWindowCluster

Examples

```
Bu <- rnorm(10)
Bv <- rnorm(10)
Alpha <- min(c(Bu, Bv))
Beta <- max(c(Bu, Bv))
CHomogeneity(Bu, Bv, Alpha, Beta, Delta = 0.5)
```

CNeighbor *Neighborhood of Time Series*

Description

This is an auxiliary function to identify which time series in Bv are E_δ^θ-neighbors of Bu, based on Definition 2 by Ciampi et al. (2010).

Usage

```
CNeighbor(Bu, Bv, Alpha, Beta, Delta, Theta)
```

Arguments

Bu	a time series vector for which the neighborhood is investigated.
Bv	a time series vector (of the same length as Bu) or a matrix (time series in columns) containing potential neighbors.
Alpha	lower limit of the time series domain.
Beta	upper limit of the time series domain.
Delta	closeness parameter, a real value in [0,1].
Theta	connectivity parameter, a real value in [0,1].

Value

A vector of logical values indicating which time series in Bv are E_δ^θ-neighbors of Bu.

Author(s)

Vyacheslav Lyubchich

References

Ciampi, A., Appice, A., and Malerba, D. (2010). Discovering trend-based clusters in spatially distributed data streams. In *International Workshop of Mining Ubiquitous and Social Environments*, pages 107–122.

See Also

CHomogeneity, CExpandSlideCluster, CSlideCluster, CExpandWindowCluster, CWindowCluster

Examples

```
Bu <- rnorm(10)
Bv <- rnorm(10)
Alpha <- min(c(Bu, Bv))
Beta <- max(c(Bu, Bv))
CNeighbor(Bu, Bv, Alpha, Beta, Delta = 0.5, Theta = 0.8)
```

`CSlideCluster` *Slide-Level Time Series Clustering*

Description

This function clusters time series at a slide level, based on the Algorithm 1 of Ciampi et al. (2010).

Usage

```
CSlideCluster(X, Alpha = NULL, Beta = NULL, Delta = NULL, Theta = 0.8)
```

Arguments

`X`	a matrix of time series observed within a slide (time series in columns).
`Alpha`	lower limit of the time series domain. Default is `quantile(X)[2] - 1.5*(quantile(X)[4] - quantile(X)[2])`.
`Beta`	upper limit of the time series domain. Default is `quantile(X)[2] + 1.5*(quantile(X)[4] - quantile(X)[2])`.
`Delta`	closeness parameter, a real value in [0,1]. Default is `0.1*(Beta - Alpha)`.
`Theta`	connectivity parameter, a real value in [0,1]. Default is 0.8.

Value

A vector of length `dim(X)[2]` with cluster labels.

Author(s)

Vyacheslav Lyubchich

References

Ciampi, A., Appice, A., and Malerba, D. (2010). Discovering trend-based clusters in spatially distributed data streams. In *International Workshop of Mining Ubiquitous and Social Environments*, pages 107–122.

See Also

`CNeighbor`, `CHomogeneity`, `CExpandSlideCluster`, `CExpandWindowCluster`, `CWindowCluster`

Examples

```
set.seed(123)
X <- matrix(rnorm(50), 10, 5)
CSlideCluster(X)
```

CWindowCluster *Window-Level Time Series Clustering*

Description

This function clusters time series at a window level, based on the Algorithm 2 of Ciampi et al. (2010).

Usage

```
CWindowCluster(X, Alpha = NULL, Beta = NULL, Delta = NULL, Theta = 0.8,
  p, w, s, Epsilon = 1)
```

Arguments

X	a matrix of time series to be clustered (time series in columns).
Alpha	lower limit of the time series domain, passed to CSlideCluster.
Beta	upper limit of the time series domain, passed to CSlideCluster.
Delta	closeness parameter, passed to CSlideCluster.
Theta	connectivity parameter, passed to CSlideCluster.
p	number of layers (time series observations) in each slide.
w	number of slides in each window.
s	step to shift a window, calculated in number of slides. Recommended values are 1 (overlapping windows) or w (non-overlapping windows).
Epsilon	a real value in [0,1] used to identify each pair of time series that are clustered together over at least w*Epsilon slides within a window (see Definition 7 by Ciampi et al., 2010). Default is 1.

Details

This is the upper-level function for time series clustering. It exploits the function CSlideCluster to cluster time series within each slide based on closeness and homogeneity measures. Then, it uses slide-level cluster assignments to cluster time series within each window.

The total length of time series (number of levels, i.e., nrow(X)) should be divisible by p.

Value

A vector (if X contains only one window) or matrix with cluster labels for each time series (columns) and window (rows).

Author(s)

Vyacheslav Lyubchich

References

Ciampi, A., Appice, A., and Malerba, D. (2010). Discovering trend-based clusters in spatially distributed data streams. In *International Workshop of Mining Ubiquitous and Social Environments*, pages 107–122.

See Also

CNeighbor, CHomogeneity, CExpandSlideCluster, CExpandWindowCluster, CSlideCluster

Examples

```
#For example, weekly data come in slides of 4 weeks
p <- 4 #number of layers in each slide (data come in a slide)

#We want to analyze the trend clusters within a window of 1 year
w <- 13 #number of slides in each window
s <- w  #step to shift a window

#Simulate 26 autoregressive time series with two years of weekly data (52*2 weeks),
#with a 'burn-in' period of 300.
N <- 26
T <- 2*p*w

set.seed(123)
phi <- c(0.5) #parameter of autoregression
X <- sapply(1:N, function(x) arima.sim(n = T + 300,
  list(order = c(length(phi), 0, 0), ar = phi)))[301:(T + 300),]
colnames(X) <- paste("TS", c(1:dim(X)[2]), sep = "")

tmp <- CWindowCluster(X, Delta = NULL, Theta = 0.8, p = p, w = w, s = s, Epsilon = 1)

#Time series were simulated with the same parameters, but based on the clustering parameters,
#not all time series join the same cluster. We can plot the main cluster for each window, and
#time series out of the cluster:
par(mfrow = c(2, 2))
ts.plot(X[c(1:(p*w)), tmp[1,] == 1], ylim = c(-4, 4),
  main = "Time series cluster 1 in window 1")
ts.plot(X[c(1:(p*w)), tmp[1,] != 1], ylim = c(-4, 4),
  main = "The rest of the time series in window 1")
ts.plot(X[c(1:(p*w)) + s*p, tmp[2,] == 1], ylim = c(-4, 4),
  main = "Time series cluster 1 in window 2")
ts.plot(X[c(1:(p*w)) + s*p, tmp[2,] != 1], ylim = c(-4, 4),
  main = "The rest of the time series in window 2")
```

DR *Downhill Riding (DR) Procedure*

Description

Downhill Riding procedure for selecting optimal tuning parameters in clustering algorithms, using an (in)stability probe.

Usage

```
DR(X, method, minPts = 3, theta = 0.9, B = 500, lb = -30, ub = 10)
```

Arguments

X	a $n \times k$ matrix where columns are k objects to be clustered, and each object contains n observations (objects could be a set of time series).
method	the clustering method to be used (currently either "TRUST" or "DBSCAN"). If method is DBSCAN, then set MinPts and optimal ϵ is selected using DR. If method is TRUST, then set theta and optimal δ is selected using DR.
minPts	the minimum number of samples in an ϵ-neighborhood of a point to be considered as a core point. The minPts is to be used only with DBSCAN method. Default value is 3.
theta	connectivity parameter $\theta \in (0, 1)$, which is to be used only with TRUST method. Default value is 0.9.
B	number of random splits in calculating the Average Cluster Deviation (ACD). Default value is 500.
lb, ub	end points for a range of search for the optimal parameter.

Value

A list containing the following components:

P_opt	the value of optimal parameter. If method is DBSCAN, then P_opt is optimal ϵ. If method is TRUST, then P_opt is optimal δ.
ACD_matrix	a matrix that returns ACD for different values of a tuning parameter. If method is DBSCAN, then the tuning parameter is ϵ. If method is TRUST, then the tuning parameter is δ.

Note

Parameters lb,ub are end points for a range of search for the optimal parameter. The parameter candidates are calculated in a way such that $P := 1.1^x, x \in lb, lb + 0.5, lb + 1.0, ..., ub$. Although the default range of search is sufficiently wide, in some cases lb,ub can be further extended if a warning message is given.

For more discussion on properties of the considered clustering algorithms and DR procedure see Huang et al. (2016).

Author(s)

Xin Huang, Yulia R. Gel

References

Ciampi, A., Appice, A., and Malerba, D. (2010). Discovering trend-based clusters in spatially distributed data streams. In *International Workshop of Mining Ubiquitous and Social Environments*, pages 107–122.

Ester, M., Kriegel, H. P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD 1996* (Vol. 96, No. 34, pp. 226–231).

Huang, X., Iliev, I., Brenning, A., and Gel, Y. R. (2016). Space-Time Clustering with Stability Probe while Riding Downhill. In *KDD 2016 workshop on Mining and Learning from Time Series (MiLeTS 2016)* http://www-bcf.usc.edu/~liu32/milets16/#papers.

See Also

CSlideCluster, dbscan

Examples

```
## Not run:
## example 1
## use iris data to test DR procedure

data(iris)
require(clue)  # calculate NMI to compare the clustering result with the ground truth
require(scatterplot3d)

Data <- scale(iris[,-5])
ground_truth_label <- iris[,5]

# perform DR procedure to select optimal eps for DBSCAN
# and save it in variable eps_opt
eps_opt <- DR(t(Data), method="DBSCAN", minPts = 5)$P_opt

# apply DBSCAN with the optimal eps on iris data
# and save the clustering result in variable res
res <- dbscan(Data, eps = eps_opt, minPts =5)$cluster

# calculate NMI to compare the clustering result with the ground truth label
clue::cl_agreement(as.cl_partition(ground_truth_label),
            as.cl_partition(as.numeric(res)), method = "NMI")
# visualize the clustering result and compare it with the ground truth result
# 3D visualization of clustering result using variables Sepal.Width, Sepal.Length,
# and Petal.Length
scatterplot3d(Data[,-4],color = res)
# 3D visualization of ground truth result using variables Sepal.Width, Sepal.Length,
# and Petal.Length
scatterplot3d(Data[,-4],color = as.numeric(ground_truth_label))


## example 2
## use synthetic time series data to test DR procedure

require(funtimes)
```

```
require(clue)
require(zoo)

# simulate 16 time series for 4 clusters, each cluster contains 4 time series
set.seed(114)
samp_Ind <- sample(12,replace=F)
time_points <- 30
X <- matrix(0,nrow=time_points,ncol = 12)
cluster1 <- sapply(1:4,function(x) arima.sim(list(order=c(1,0,0),ar=c(0.2)),
                                             n=time_points,mean=0,sd=1))
cluster2 <- sapply(1:4,function(x) arima.sim(list(order=c(2,0,0),ar=c(0.1,-0.2)),
                                             n=time_points,mean=2,sd=1))
cluster3 <- sapply(1:4,function(x) arima.sim(list(order=c(1,0,1),ar=c(0.3),ma=c(0.1)),
                                             n=time_points,mean=6,sd=1))

X[,samp_Ind[1:4]] <- t(round(cluster1,4))
X[,samp_Ind[5:8]] <- t(round(cluster2,4))
X[,samp_Ind[9:12]] <- t(round(cluster3,4))


# create ground truth label of the synthetic data
ground_truth_label = matrix(1,nrow=12,ncol=1)
for(k in 1:3){
  ground_truth_label[samp_Ind[(4*k-4+1):(4*k)]] = k
}

# perform DR procedure to select optimal delta for TRUST
# and save it in variable delta_opt
delta_opt <- DR(X,method="TRUST")$P_opt

# apply TRUST with the optimal delta on the synthetic data
# and save the clustering result in variable res
res <- CSlideCluster(X,Delta=delta_opt ,Theta=0.9)

# calculate NMI to compare the clustering result with the ground truth label
clue::cl_agreement(as.cl_partition(as.numeric(ground_truth_label)),
             as.cl_partition(as.numeric(res)),method = "NMI")

# visualize the clustering result and compare it with the ground truth result
# visualization of the clustering result obtained by TRUST
plot.zoo(X, type = "l",plot.type = "single",col = res, xlab = "Time Index", ylab ="")
# visualization of the ground truth result
plot.zoo(X, type = "l",plot.type = "single",col = ground_truth_label,
         xlab = "Time Index", ylab ="")



## End(Not run)
```

HVK *HVK Estimator*

Description

Estimates coefficients in non-parametric autoregression using the difference-based approach by Hall and Van Keilegom (2003).

Usage

```
HVK(X, m1 = NULL, m2 = NULL, ar.order = 1)
```

Arguments

`X`	univariate time series. Missing values are not allowed.
`m1, m2`	subsidiary smoothing parameters. Default `m1 = round(length(X)^(0.1))`, `m2 = round(length(X)^(0.5))`.
`ar.order`	order of the non-parametric autoregression (specified by user).

Details

First, autocovariances are estimated (formula (2.6) by Hall and Van Keilegom, 2003):

$$\hat{\gamma}(0) = \frac{1}{m_2 - m_1 + 1} \sum_{m=m_1}^{m_2} \frac{1}{2(n-m)} \sum_{i=m+1}^{n} \{(D_m X)_i\}^2,$$

$$\hat{\gamma}(j) = \hat{\gamma}(0) - \frac{1}{2(n-j)} \sum_{i=j+1}^{n} \{(D_j X)_i\}^2,$$

where n=`length(X)` is sample size, D_j is a difference operator such that $(D_j X)_i = X_i - X_{i-j}$. Then, Yule-Walker method is used to derive autoregression coefficients.

Value

Vector of length `ar.order` with estimated autoregression coefficients.

Author(s)

Yulia R. Gel, Vyacheslav Lyubchich, Xingyu Wang

References

Hall, P. and Van Keilegom, I. (2003). Using difference-based methods for inference in nonparametric regression with time series errors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65: 443–456. DOI: 10.1111/1467-9868.00395

Examples

```
X <- arima.sim(n = 300, list(order = c(1, 0, 0), ar = c(0.6)))
HVK(as.vector(X), ar.order = 1)
```

`i.tails` *Interval-Based Tails Comparison*

Description

This function compares right tails of two sample distributions using an interval-based approach (IBA).

Usage

```
i.tails(x0, x1, d = NULL)
```

Arguments

`x0,x1`	vectors of the same length (preferably). Tail in `x1` is compared against the tail in `x0`.
`d`	a threshold defining the tail. The threshold is the same for both `x0` and `x1`. Default is `quantile(x0, probs = 0.99)`.

Details

Sturges' formula is used to calculate number of intervals (k) for `x0` $\geq$ `d`, then interval width is derived. The tails, `x0` $\geq$ `d` and `x1` $\geq$ `d`, are divided into the intervals. Number of `x1`-values within each interval is compared with the number of `x0`-values within the same interval (this difference is reported as `Nk`).

Value

A list with two elements:

`Nk`	vector that tells how many more `x1`-values compared with `x0`-values there are within each interval.
`Ck`	vector of intervals' centers.

Author(s)

Calvin Chu, Yulia R. Gel, Vyacheslav Lyubchich

References

Chu, C., Gel, Y. R., and Lyubchich, V. (2015). Climate change from an insurance perspective: a case study of Norway. In J. G. Dy et al. (Eds.) *Proceedings of the 5th International Workshop on Climate Informatics: CI2015*. September 24–25, 2015, Boulder, Colorado, USA.

Lyubchich, V. and Gel, Y. R. (2017). Can we weather proof our insurance? *Environmetrics* 28(2): e2433. DOI: 10.1002/env.2433

See Also

q.tails

Examples

```
x0 <- rnorm(1000)
x1 <- rt(1000, 5)
i.tails(x0, x1)
```

---

notrend.test *Sieve Bootstrap Based Test for the Null Hypothesis of no Trend*

Description

A combination of time series trend tests for testing the null hypothesis of no trend, versus the alternative hypothesis of a linear trend (Student's t-test), or monotonic trend (Mann-Kendall test), or possibly non-monotonic trend (WAVK test).

Usage

```
notrend.test(x, B = 1000, test = c("t", "MK", "WAVK"),
             ar.method = "HVK", ar.order = NULL, BIC = TRUE,
             factor.length = c("user.defined", "adaptive.selection"),
             Window = NULL, q = 3/4, j = c(8:11))
```

Arguments

x	univariate time series. Missing values are not allowed.
B	number of bootstrap simulations to obtain empirical critical values.
test	trend test to implement: Student's t-test ("t", default), Mann-Kendall test ("MK"), or WAVK test ("WAVK", see WAVK).
ar.method	method of estimating autoregression coefficients. Default "HVK" delivers robust difference-based estimates by Hall and Van Keilegom (2003). Alternatively, options of ar function can be used, such as "burg", "ols", "mle", and "yw".
ar.order	order of autoregressive filter when BIC = FALSE, or the maximal order for BIC-based filtering. Default is round(10*log10(length(x))).
BIC	logical value indicates whether the order of autoregressive filter should be selected by Bayesian information criterion (BIC). If TRUE (default), models of orders 0,1,...,ar.order or 0,1,...,round(10*log10(length(x))) are considered, depending on whether ar.order is defined or not.
factor.length	method to define the length of local windows (factors). Used only if test = "WAVK". Default option "user.defined" allows to set only one value of the argument Window. The option "adaptive.selection" sets method = "boot" and employs heuristic m-out-of-n subsampling algorithm (Bickel and Sakov, 2008) to select an optimal window from the set of possible windows length(x)*q^j whose values are mapped to the largest previous integer and greater than 2. x is the time series tested.

`Window`	length of the local window (factor), default is `round(0.1*length(x))`. Used only if `test = "WAVK"`. This argument is ignored if `factor.length = "adaptive.selection"`.
`q`	scalar from 0 to 1 to define the set of possible windows when `factor.length = "adaptive.selection"`. Used only if `test = "WAVK"`. Default is $3/4$. This argument is ignored if `factor.length = "user.defined"`.
`j`	numeric vector to define the set of possible windows when `factor.length = "adaptive.selection"`. Used only if `test = "WAVK"`. Default is `c(8:11)`. This argument is ignored if `factor.length = "user.defined"`.

Details

The current function tests the null hypothesis of no trend. For more general alternative shapes of the trend, use `wavk.test`. Notice that `wavk.test` employs hybrid bootstrap, which is alternative to the sieve bootstrap employed by the current function.

Value

A list with class htest containing the following components:

`method`	name of the method.
`data.name`	name of the data.
`statistic`	value of the test statistic.
`p.value`	p-value of the test.
`alternative`	alternative hypothesis.
`estimate`	list with the following elements: employed AR order and estimated AR coefficients.
`parameter`	window that was used in WAVK test, included in the output only if `test = "WAVK"`.

Author(s)

Yulia R. Gel, Vyacheslav Lyubchich

References

Bickel, P. J. and Sakov, A. (2008). On the choice of m in the m out of n bootstrap and confidence bounds for extrema. *Statistica Sinica* 18: 967–985.

Hall, P. and Van Keilegom, I. (2003). Using difference-based methods for inference in nonparametric regression with time series errors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65: 443–456. DOI: 10.1111/1467-9868.00395

Lyubchich, V., Gel, Y. R., and El-Shaarawi, A. (2013). On detecting non-monotonic trends in environmental time series: a fusion of local regression and bootstrap. *Environmetrics* 24: 209–226. DOI: 10.1002/env.2212

Wang, L., Akritas, M. G., and Van Keilegom, I. (2008). An ANOVA-type nonparametric diagnostic test for heteroscedastic regression models. *Journal of Nonparametric Statistics* 20(5): 365–382. DOI: 10.1080/10485250802066112

Wang, L. and Van Keilegom, I. (2007). Nonparametric test for the form of parametric regression with time series errors. *Statistica Sinica* 17: 369–386.

See Also

ar, HVK, WAVK, wavk.test

Examples

```
## Not run:
# Fix seed for reproducible simulations:
set.seed(1)

#Simulate autoregressive time series of length n with smooth linear trend:
n <- 200
tsTrend <- 1 + 2*(1:n/n)
tsNoise <- arima.sim(n = n, list(order = c(2, 0, 0), ar = c(0.5, -0.1)))
U <- tsTrend + tsNoise
plot.ts(U)


#Use t-test
notrend.test(U)

#Use Mann-Kendall test and Yule-Walker estimates of the AR parameters
notrend.test(U, test = "MK", ar.method = "yw")


#Use WAVK test for the H0 of no trend, with m-out-of-n selection of the local window:
notrend.test(U, test = "WAVK", factor.length = "adaptive.selection")
# Sample output:
## Sieve-bootstrap WAVK trend test
##
##data:  U
##WAVK test statistic = 21.654, moving window = 15, p-value < 2.2e-16
##alternative hypothesis: (non-)monotonic trend.
##sample estimates:
##$AR_order
##[1] 1
##
##$AR_coefficients
##    phi_1
##0.4041848

## End(Not run)
```

---

purity *Clustering Purity*

Description

Calculate purity of the clustering results.

Usage

```
purity(classes, clusters)
```

Arguments

`classes, clusters`
: vectors of equal lengths, with labels of underlying true `classes` and assigned `clusters`. Number of unique elements in `classes` and `clusters` may differ.

Details

Following Manning et al. (2008), each cluster is assigned to the class which is most frequent in the cluster, then

$$Purity(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|,$$

where $\Omega = \{\omega_1, \ldots, \omega_K\}$ is the set of identified clusters and $C = \{c_1, \ldots, c_J\}$ is the set of classes. That is, within each class $j = 1, \ldots, J$ find the size of the most populous cluster from the $K - j$ unassigned clusters. Then, sum together the $\min(K, J)$ sizes found and divide by N, where N = `length(classes)` = `length(clusters)`.

If $\max_j |\omega_k \cap c_j|$ is not unique for some j, it is assigned to the class which second maximum is the smallest, to maximize the $Purity$ (see 'Examples').

Value

A list with two elements:

`pur`
: purity value.

`out`
: table with $\min(K, J)$ = `min(length(unique(classes)), length(unique(clusters)))` rows and the following columns: `ClassLabels`, `ClusterLabels`, and `ClusterSize`.

Author(s)

Vyacheslav Lyubchich

References

Manning, C. D., Raghavan, P., and Schutze, H. (2008). *Introduction to Information Retrieval*. New York: Cambridge University Press.

Schaeffer, E. D., Testa, J. M., Gel, Y. R., and Lyubchich, V. (2016). On information criteria for dynamic spatio-temporal clustering. In A. Banerjee et al. (Eds.) *Proceedings of the 6th International Workshop on Climate Informatics: CI 2016*. NCAR Technical Note NCAR/TN-529+PROC, September 2016, pages 5–8. DOI: 10.5065/D6K072N6

Examples

```
# Fix seed for reproducible simulations:
set.seed(1)

##### Example 1
#Create some classes and cluster labels:
classes <- rep(LETTERS[1:3], each = 5)
clusters <- sample(letters[1:5], length(classes), replace = TRUE)

#From the table below:
# - cluster 'b' corresponds to class A;
# - either of the clusters 'd' and 'e' can correspond to class B,
#   however, 'e' should be chosen, because cluster 'd' also highly
#   intersects with Class C. Thus,
# - cluster 'd' corresponds to class C.
table(classes, clusters)
##       clusters
##classes a b c d e
##      A 0 3 1 0 1
##      B 1 0 0 2 2
##      C 1 2 0 2 0

#The function does this choice automatically:
purity(classes, clusters)

#Sample output:
##$pur
##[1] 0.4666667
##
##$out
##  ClassLabels ClusterLabels ClusterSize
##1           A             b           3
##2           B             e           2
##3           C             d           2


##### Example 2
#The labels can be also numeric:
classes <- rep(1:5, each = 3)
clusters <- sample(1:3, length(classes), replace = TRUE)
purity(classes, clusters)
```

`q.tails` *Quantile-Based Tails Comparison*

Description

This function compares right tails of two sample distributions using a quantile-based approach (QBA).

Usage

```
q.tails(x0, x1, q = 0.99)
```

Arguments

x0,x1	vectors of the same length (preferably). Tail in x1 is compared against the tail in x0.
q	a quantile defining the right tail for both x0 and x1. Values above the thresholds quantile(x0, probs = q) and quantile(x1, probs = q) are considered as the respective right tails.

Details

Sturges' formula is used to calculate number of intervals (k) to split the upper $100(1-q)\%$ portion of x0 and x1 (the right tails). Then, each tail is divided into equally-filled intervals with a quantile step $d = (1-q)/k$. Pk reports the difference between corresponding intervals' centers obtained from x0 and x1.

Value

A list with two elements:

d	the quantile step.
Pk	vector of differences of the intervals' centers.

Author(s)

Vyacheslav Lyubchich, Yulia R. Gel

References

Lyubchich, V. and Gel, Y. R. (2017). Can we weather proof our insurance? *Environmetrics* 28(2): e2433. DOI: 10.1002/env.2433

Soliman, M., Lyubchich, V., Gel, Y. R., Naser, D., and Esterby, S. (2015). Evaluating the impact of climate change on dynamics of house insurance claims. Ch. 16 in V. Lakshmanan et al. (Eds.) *Machine Learning and Data Mining Approaches to Climate Science*, pp. 175–183. Springer International Publishing. DOI: 10.1007/978-3-319-17220-0_16

Soliman, M., Naser, D., Lyubchich, V., Gel, Y. R., and Esterby, S. (2014). Evaluating the impact of climate change on dynamics of house insurance claims. In *Proceedings of the 4th International Workshop on Climate Informatics: CI2014*. September 25–26, 2014, Boulder, Colorado, USA.

See Also

i.tails

Examples

```
x0 <- rnorm(1000)
x1 <- rt(1000, 5)
q.tails(x0, x1)
```

`sync.test` *Time Series Trend Synchronism Test*

Description

Non-parametric test for synchronism of parametric trends in multiple time series. The method tests whether N observed time series exhibit the same trend of some pre-specified smooth parametric form.

Usage

```
sync.test(formula, B = 1000, Window = NULL, q = NULL, j = NULL,
          ar.order = NULL, ar.method = "HVK", BIC = TRUE)
```

Arguments

`formula`	an object of class "formula", specifying the form of the common parametric time trend to be tested in a T by N matrix of time series (time series in columns). Variable t should be used to specify the form of the trend, where t is specified within the function as a regular sequence on the interval (0,1]. See 'Examples'.
`B`	number of bootstrap simulations to obtain empirical critical values.
`Window`	scalar or N-vector with lengths of the local windows (factors). If only one value is set, the same `Window` is applied to each time series. N-vector specifies a particular window for each time series. If no `Window` is specified, the automatic algorithm for optimal window selection is performed as a default option (see 'Details').
`q`	scalar from 0 to 1 to define the set of possible windows `T*q^j` and to automatically select an optimal window for each time series. Default is $3/4$. This argument is ignored if `Window` is set by user.
`j`	numeric vector to define the set of possible windows `T*q^j` and to automatically select an optimal window for each time series. Default is `c(8:11)`. This argument is ignored if `Window` is set by user.
`ar.order`	order of autoregressive filter when `BIC = FALSE`, or the maximal order for BIC-based filtering. Default is `round(10*log10(T))`. The `ar.order` can be a scalar or N-vector. If scalar, the same `ar.order` is applied to each time series. N-vector specifies a separate `ar.order` for each time series.
`ar.method`	method of estimating autoregression coefficients. Default `"HVK"` delivers robust difference-based estimates by Hall and Van Keilegom (2003). Alternatively, options of `ar` function can be used, such as `"burg"`, `"ols"`, `"mle"`, and `"yw"`.

`BIC` logical value indicates whether the order of autoregressive filter should be selected by Bayesian information criterion (BIC). If `TRUE` (default), models of orders 0,1,...,`ar.order` or 0,1,...,`round(10*log10(T))` are considered, depending on whether `ar.order` is defined or not.

Details

Arguments `Window`, `j`, and `q` are used to set windows for the local regression. Current version of the function assumes two options: (1) user specifies one fixed window for each time series using the argument `Window` (if `Window` is set, `j` and `q` are ignored), and (2) user specifies a set of windows by `j` and `q` to apply this set to each time series and to select an optimal window using a heuristic m-out-of-n subsampling algorithm (Bickel and Sakov, 2008). The option of selecting windows automatically for some of the time series, while for other time series the window is fixed, is not available yet. If none of these three arguments is set, default `j` and `q` are used. Values `T*q^j` are mapped to the largest previous integer, then only those greater than 2 are used.

Value

A list of class htest containing the following components:

`method` name of the method.

`data.name` name of the data.

`statistic` value of the test statistic.

`p.value` p-value of the test.

`alternative` alternative hypothesis.

`estimate` list with elements `common_trend_estimates`, `ar_order_used`, `Window_used`, `wavk_obs`, and `all_considered_windows`. The latter is a table with bootstrap and asymptotic test results for all considered windows, i.e., without adaptive selection of the local window.

Author(s)

Yulia R. Gel, Vyacheslav Lyubchich, Ethan Schaeffer, Xingyu Wang

References

Bickel, P. J. and Sakov, A. (2008). On the choice of m in the m out of n bootstrap and confidence bounds for extrema. *Statistica Sinica* 18: 967–985.

Lyubchich, V. and Gel, Y. R. (2016). A local factor nonparametric test for trend synchronism in multiple time series. *Journal of Multivariate Analysis* 150: 91–104. DOI: 10.1016/j.jmva.2016.05.004

Lyubchich, V., Gel, Y. R., and El-Shaarawi, A. (2013). On detecting non-monotonic trends in environmental time series: a fusion of local regression and bootstrap. *Environmetrics* 24: 209–226. DOI: 10.1002/env.2212

Wang, L., Akritas, M. G., and Van Keilegom, I. (2008). An ANOVA-type nonparametric diagnostic test for heteroscedastic regression models. *Journal of Nonparametric Statistics* 20(5): 365–382. DOI: 10.1080/10485250802066112

Wang, L. and Van Keilegom, I. (2007). Nonparametric test for the form of parametric regression with time series errors. *Statistica Sinica* 17: 369–386.

See Also

ar, HVK, WAVK, wavk.test

Examples

```
# Fix seed for reproducible simulations:
set.seed(1)

# Simulate two autoregressive time series of length n without trend
#(i.e., with zero or constant trend)
# and arrange the series into a matrix:
n <- 200
y1 <- arima.sim(n = n, list(order = c(1, 0, 0), ar = c(0.6)))
y2 <- arima.sim(n = n, list(order = c(1, 0, 0), ar = c(-0.2)))
Y <- cbind(y1, y2)
plot.ts(Y)


#Test H0 of a common linear trend:
## Not run:
sync.test(Y ~ t, B = 500)
## End(Not run)
# Sample output:
## Non-parametric test for synchronism of parametric trends
##
##data:  Y
##Test statistic = -0.0028999, p-value = 0.7
##alternative hypothesis: common trend is not of the form Y ~ t.
##sample estimates:
##$common_trend_estimates
##               Estimate Std. Error    t value  Pr(>|t|)
##(Intercept) -0.02472566  0.1014069 -0.2438261 0.8076179
##t            0.04920529  0.1749859  0.2811958 0.7788539
##
##$ar.order_used
##         y1 y2
##ar.order  1  1
##
##$Window_used
##       y1 y2
##Window 15  8
##
##$all_considered_windows
## Window    Statistic p-value Asympt. p-value
##      8 -0.000384583   0.728       0.9967082
##     11 -0.024994408   0.860       0.7886005
##     15 -0.047030164   0.976       0.6138976
##     20 -0.015078579   0.668       0.8714980
##
##$wavk_obs
##[1]  0.05827148 -0.06117136
```

```
# Add a time series y3 with a different linear trend and re-apply the test:
y3 <- 1 + 3*((1:n)/n) + arima.sim(n = n, list(order = c(1, 0, 0), ar = c(-0.2)))
Y2 <- cbind(Y, y3)
plot.ts(Y2)
## Not run:
sync.test(Y2 ~ t, B = 500)
## End(Not run)
# Sample output:
## Non-parametric test for synchronism of parametric trends
##
##data:  Y2
##Test statistic = 0.48579, p-value < 2.2e-16
##alternative hypothesis: common trend is not of the form Y2 ~ t.
##sample estimates:
##$common_trend_estimates
##              Estimate Std. Error  t value     Pr(>|t|)
##(Intercept) -0.3632963 0.07932649 -4.57976 8.219360e-06
##t            0.7229777 0.13688429  5.28167 3.356552e-07
##
##$ar.order_used
##         Y.y1 Y.y2 y3
##ar.order    1    1  0
##
##$Window_used
##       Y.y1 Y.y2 y3
##Window    8   11  8
##
##$all_considered_windows
## Window Statistic p-value Asympt. p-value
##      8 0.4930069       0    1.207378e-05
##     11 0.5637067       0    5.620248e-07
##     15 0.6369703       0    1.566057e-08
##     20 0.7431621       0    4.201484e-11
##
##$wavk_obs
##[1]  0.08941797 -0.07985614  0.34672734

#Other hypothesized trend forms can be specified, for example:
## Not run:
sync.test(Y ~ 1) #constant trend
sync.test(Y ~ poly(t, 2)) #quadratic trend
sync.test(Y ~ poly(t, 3)) #cubic trend
## End(Not run)
```

WAVK *WAVK Statistic*

Description

Computes statistic for testing the parametric form of a regression function, suggested by Wang, Akritas and Van Keilegom (2008).

Usage

```
WAVK(z, kn = NULL)
```

Arguments

z	pre-filtered univariate time series (see formula (2.1) by Wang and Van Keilegom, 2007): $$Z_i = \left(Y_{i+p} - \sum_{j=1}^{p} \hat{\phi}_{j,n} Y_{i+p-j} \right) - \left(f(\hat{\theta}, t_{i+p}) - \sum_{j=1}^{p} \hat{\phi}_{j,n} f(\hat{\theta}, t_{i+p-j}) \right),$$ where Y_i is observed time series of length n, $\hat{\theta}$ is an estimator of hypothesized parametric trend $f(\theta, t)$, and $\hat{\phi}_p = (\hat{\phi}_{1,n}, \ldots, \hat{\phi}_{p,n})'$ are estimated coefficients of an autoregressive filter of order p. Missing values are not allowed.
kn	length of the local window.

Value

A list with following components:

Tn	test statistic based on artificial ANOVA and defined by Wang and Van Keilegom (2007) as a difference of mean square for treatments (MST) and mean square for errors (MSE): $$T_n = MST - MSE = \frac{k_n}{n-1} \sum_{t=1}^{T} \left(\overline{V}_{t.} - \overline{V}_{..} \right)^2 - \frac{1}{n(k_n - 1)} \sum_{t=1}^{n} \sum_{j=1}^{k_n} \left(V_{tj} - \overline{V}_{t.} \right)^2,$$ where $\{V_{t1}, \ldots, V_{tk_n}\} = \{Z_j : j \in W_t\}$, W_t is a local window, $\overline{V}_{t.}$ and $\overline{V}_{..}$ are the mean of the tth group and the grand mean, respectively.
Tns	standardized version of Tn according to Theorem 3.1 by Wang and Van Keilegom (2007): $$T_{ns} = \left(\frac{n}{k_n} \right)^{\frac{1}{2}} T_n \Big/ \left(\frac{4}{3} \right)^{\frac{1}{2}} \sigma^2,$$ where n is length and σ^2 is variance of the time series. Robust difference-based Rice's estimator (Rice, 1984) is used to estimate σ^2.
p.value	p-value for Tns based on its asymptotic $N(0, 1)$ distribution.

Author(s)

Yulia R. Gel, Vyacheslav Lyubchich

References

Rice, J. (1984). Bandwidth choice for nonparametric regression. *The Annals of Statistics* 12: 1215–1230. DOI: 10.1214/aos/1176346788

Wang, L., Akritas, M. G., and Van Keilegom, I. (2008). An ANOVA-type nonparametric diagnostic test for heteroscedastic regression models. *Journal of Nonparametric Statistics* 20(5): 365–382. DOI: 10.1080/10485250802066112

Wang, L. and Van Keilegom, I. (2007). Nonparametric test for the form of parametric regression with time series errors. *Statistica Sinica* 17: 369–386.

See Also

`wavk.test`

Examples

```
z <- rnorm(300)
WAVK(z, kn = 7)
```

`wavk.test` *WAVK Trend Test*

Description

Non-parametric test to detect (non-)monotonic parametric trends in time series.

Usage

```
wavk.test(formula, factor.length = c("user.defined", "adaptive.selection"),
        Window = NULL, q = 3/4, j = c(8:11), B = 1000, method = c("boot", "asympt"),
          ar.order = NULL, ar.method = "HVK", BIC = TRUE, out = FALSE)
```

Arguments

`formula`	an object of class "`formula`", specifying the form of the parametric time trend to be tested. Variable t should be used to specify the form, where t is specified within the function as a regular sequence on the interval (0,1]. See 'Examples'.
`factor.length`	method to define the length of local windows (factors). Default option "`user.defined`" allows to set only one value of the argument `Window`. The option "`adaptive.selection`" sets `method = "boot"` and employs heuristic m-out-of-n subsampling algorithm (Bickel and Sakov, 2008) to select an optimal window from the set of possible windows `length(x)*q^j` whose values are mapped to the largest previous integer and greater than 2. `x` is the time series tested.
`Window`	length of the local window (factor), default is `round(0.1*length(x))`, where `x` is the time series tested. This argument is ignored if `factor.length = "adaptive.selection"`.

q	scalar from 0 to 1 to define the set of possible windows when factor.length = "adaptive.selection". Default is $3/4$. This argument is ignored if factor.length = "user.defined".
j	numeric vector to define the set of possible windows when factor.length = "adaptive.selection". Default is c(8:11). This argument is ignored if factor.length = "user.defined".
B	number of bootstrap simulations to obtain empirical critical values.
method	method of obtaining critical values: from asymptotical ("asympt") or bootstrap ("boot") distribution. If factor.length = "adaptive.selection" the option "boot" is used.
ar.order	order of autoregressive filter when BIC = FALSE, or the maximal order for BIC-based filtering. Default is round(10*log10(length(x))), where x is the time series tested.
ar.method	method of estimating autoregression coefficients. Default "HVK" delivers robust difference-based estimates by Hall and Van Keilegom (2003). Alternatively, options of ar function can be used, such as "burg", "ols", "mle", and "yw".
BIC	logical value indicates whether the order of autoregressive filter should be selected by Bayesian information criterion (BIC). If TRUE (default), models of orders 0,1,...,ar.order or 0,1,...,round(10*log10(length(x))) are considered, depending on whether ar.order is defined or not. x is the time series tested.
out	logical value indicates whether full output should be shown. Default is FALSE.

Value

A list with class htest containing the following components:

method	name of the method.
data.name	name of the data.
statistic	value of the test statistic.
p.value	p-value of the test.
alternative	alternative hypothesis.
parameter	window that was used.
estimate	list with the following elements: estimated trend coefficients; user-defined or BIC-selected AR order; estimated AR coefficients; and, if factor.length = "adaptive.selection", test results for all considered windows.

Author(s)

Yulia R. Gel, Vyacheslav Lyubchich, Ethan Schaeffer

References

Bickel, P. J. and Sakov, A. (2008). On the choice of m in the m out of n bootstrap and confidence bounds for extrema. *Statistica Sinica* 18: 967–985.

Hall, P. and Van Keilegom, I. (2003). Using difference-based methods for inference in nonparametric regression with time series errors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65: 443–456. DOI: 10.1111/1467-9868.00395

Lyubchich, V., Gel, Y. R., and El-Shaarawi, A. (2013). On detecting non-monotonic trends in environmental time series: a fusion of local regression and bootstrap. *Environmetrics* 24: 209–226. DOI: 10.1002/env.2212

Wang, L., Akritas, M. G., and Van Keilegom, I. (2008). An ANOVA-type nonparametric diagnostic test for heteroscedastic regression models. *Journal of Nonparametric Statistics* 20(5): 365–382. DOI: 10.1080/10485250802066112

Wang, L. and Van Keilegom, I. (2007). Nonparametric test for the form of parametric regression with time series errors. *Statistica Sinica* 17: 369–386.

See Also

ar, HVK, WAVK, sync.test

Examples

```
# Fix seed for reproducible simulations:
set.seed(1)

#Simulate autoregressive time series of length n with smooth quadratic trend:
n <- 100
tsTrend <- 1 + 2*(1:n/n) + 4*(1:n/n)^2
tsNoise <- arima.sim(n = n, list(order = c(2, 0, 0), ar = c(-0.7, -0.1)))
U <- tsTrend + tsNoise
plot.ts(U)


#Test H0 of a linear trend, with m-out-of-n selection of the local window:
## Not run:
wavk.test(U ~ t, factor.length = "adaptive.selection")
## End(Not run)
# Sample output:
## Trend test by Wang, Akritas, and Van Keilegom (bootstrap p-values)
##
##data:  U
##WAVK test statistic = 5.3964, adaptively selected window = 4, p-value < 2.2e-16
##alternative hypothesis: trend is not of the form U ~ t.


#Test H0 of a quadratic trend, with m-out-of-n selection of the local window
#and output of all results:
## Not run:
wavk.test(U ~ poly(t, 2), factor.length = "adaptive.selection", out = TRUE)
## End(Not run)
# Sample output:
## Trend test by Wang, Akritas, and Van Keilegom (bootstrap p-values)
##
##data:  U
```

```
##WAVK test statistic = 0.40083, adaptively selected window = 4, p-value = 0.576
##alternative hypothesis: trend is not of the form U ~ poly(t, 2).
##sample estimates:
##$trend_coefficients
##(Intercept) poly(t, 2)1 poly(t, 2)2
##   3.408530   17.681422    2.597213
##
##$AR_order
##[1] 1
##
##$AR_coefficients
##         phi_1
##[1] -0.7406163
##
##$all_considered_windows
## Window WAVK-statistic p-value
##      4     0.40083181   0.576
##      5     0.06098625   0.760
##      7    -0.57115451   0.738
##     10    -1.02982929   0.360


# Test H0 of no trend (constant trend) using asymptotic distribution of statistic.
wavk.test(U ~ 1, method = "asympt")
# Sample output:
## Trend test by Wang, Akritas, and Van Keilegom (asymptotic p-values)
##
##data:  U
##WAVK test statistic = 25.999, user-defined window = 10, p-value < 2.2e-16
##alternative hypothesis: trend is not of the form U ~ 1.
```

Index