Package 'DriftBurstHypothesis'

April 14, 2019

Type Package

Title Calculates the Test-Statistic for the Drift Burst Hypothesis

Version 0.1.2

Date 2019-04-13

Author Emil Sjoerup

Maintainer Emil Sjoerup <emilsjoerup@live.dk>

Description Calculates the T-Statistic for the drift burst hypothesis from the working paper Christensen, Oomen and Reno (2018) <DOI:10.2139/ssrn.2842535>. The authors' MATLAB code is available upon request, see: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2842535>.

License GPL-3

BugReports https://github.com/emilsjoerup/DriftBurstHypothesis/issues

URL https://github.com/emilsjoerup/DriftBurstHypothesis

Imports Rcpp (>= 0.12.18), xts, zoo,

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-04-14 11:22:42 UTC

R topics documented:

`DriftBurstHypothesis-package` *Calculates the Test-Statistic for the Drift Burst Hypothesis*

Description

Calculates the T-Statistic for the drift burst hypothesis from the working paper Christensen, Oomen and Reno (2018) <DOI:10.2139/ssrn.2842535>. The authors' MATLAB code is available upon request, see: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2842535>.

Details

The DESCRIPTION file:

Package:	DriftBurstHypothesis
Type:	Package
Title:	Calculates the Test-Statistic for the Drift Burst Hypothesis
Version:	0.1.2
Date:	2019-04-13
Author:	Emil Sjoerup
Maintainer:	Emil Sjoerup <emilsjoerup@live.dk>
Description:	Calculates the T-Statistic for the drift burst hypothesis from the working paper Christensen, Oomen and Reno (
License:	GPL-3
BugReports:	https://github.com/emilsjoerup/DriftBurstHypothesis/issues
URL:	https://github.com/emilsjoerup/DriftBurstHypothesis
Imports:	Rcpp (>= 0.12.18), xts, zoo,
LinkingTo:	Rcpp, RcppArmadillo

Index of help topics:

Author(s)

Emil Sjoerup

Maintainer: Emil Sjoerup <emilsjoerup@live.dk>

References

Christensen, Oomen and Reno (2018) <DOI:10.2139/ssrn.2842535>.

drift_bursts *Drift Bursts*

Description

Calculates the Test-Statistic for the Drift Burst Hypothesis.

Usage

```
drift_bursts(time = NULL, logprices, testtimes = seq(34200, 57600, 60),
             PreAverage = 5, AcLag = -1L, Mean_bandwidth = 300L,
             Variance_bandwidth = 900L, bParallelize = FALSE, iCores = NA,
             warnings = TRUE)
```

Arguments

time	Either: A numeric of timestamps for the trades in seconds after midnight. Or: NULL, when the time argument is NULL, the logprices argument must be an xts object.
logprices	A numeric or xts object containing the log-prices of the asset.
testtimes	A numeric containing the times at which to calculate the tests. The standard of seq(34200, 57600, 60) denotes calculating the test-statistic once per minute, i.e. 391 times for a typical 6.5 hour trading day from 9:30:00 to 16:00:00. See details.
PreAverage	An integer denoting the period for pre-averaging estimates of the log-prices.
AcLag	An integer denoting which lag is to be used for the HAC estimator of the variance - the standard of -1 denotes using an automatic lag selection algorithm.
Mean_bandwidth	An integer denoting the bandwidth for the left-sided exponential kernel for the mean.
Variance_bandwidth	An integer denoting the bandwidth for the left-sided exponential kernel for the variance.
bParallelize	A Boolean to determine whether to parallelize the underlying C++ code. (Using OpenMP)
iCores	An integer denoting the number of cores to use for calculating the code when paralellized. If this argument is not provided, sequential evaluation will be used even though bParallelize is TRUE
warnings	A logical denoting whether warnings should be shown.

Details

********** It is important to properly handle the testtimes argument of this function. If for example the standard is passed on a half day, the program may crash if parallelised. I have not been able to find a suitable solution for this in the parallelised case. Please, if you have one, send me an e-mail! **********

The DBH test statistic cannot be calculated before 1 period of testtimes has passed.

The test statistic is unstable before `max(Mean_bandwidth , Variance_bandwidth)` seconds has passed.

If times is provided and logprices is an `xts` object, the indices of logprice will be used regardless.

Note that using an `xts` argument is slower than using a `numeric` due to the creation of the timestamps from the index of the input.

Value

A list containing the series of the drift burst hypothesis test-statistic as well as the estimated local mean and variance series.

Author(s)

Emil Sjoerup

References

Christensen, Oomen and Reno (2018) <DOI:10.2139/ssrn.2842535>.

Examples

```
#Simulate from a stochastic volatility model.
#Both a flash crash and flash rally are coded into the function.
StochasticVolatilitySim = function(iT, dSigma, dPhi, dMu){
  vSim = numeric(iT)
  vEps = rnorm(iT , sd =dSigma)
  vEpsy = rnorm(iT)
  vEps[30001:32000] = rnorm(2000 ,sd =seq(from = dSigma ,
                                          to = 2*dSigma , length.out = 2000))
  vEps[32001:34000] = rnorm(2000 ,sd =seq(from = 2*dSigma ,
                                          to = dSigma , length.out = 2000))
  vEpsy[30001:32000] = -rnorm(2000 ,mean =seq(from = 0,
                                              to = 0.3 , length.out = 2000))
  vEpsy[32001:34000] = -rnorm(2000 ,mean =seq(from = 0.3,
                                              to = 0 , length.out = 2000))


  vEps[60001:63000] = rnorm(3000,sd = seq(from = dSigma ,
                                          to = 2*dSigma , length.out = 3000))
  vEps[63001:66000] = rnorm(3000,  sd = seq(from = 2*dSigma ,
                                            to =  dSigma, length.out = 3000))

  vEpsy[60001:63000] = rnorm(3000 ,mean =seq(from = 0,
                                             to = 0.2 , length.out = 3000))
  vEpsy[63001:66000] = rnorm(3000 ,mean =seq(from = 0.2,
                                             to = 0 , length.out = 3000))
  vSim[1] = dMu + dPhi *rnorm(1 , mean = dMu , sd = dSigma /sqrt(1-dPhi^2))
  for (i in 2:iT) {
    vSim[i] = dMu + dPhi * (vSim[(i-1)] - dMu) + vEps[i]
  }
```

```
  vY = exp(vSim/2) * vEpsy
  return(vY)
}
#Set parameter values of the simulation
iT = 66500; dSigma = 0.3; dPhi = 0.98; dMu = -10;
#set seed for reproducibility
set.seed(123)
#Simulate the series
vY = 500+cumsum(StochasticVolatilitySim(iT, dSigma, dPhi, dMu))

#insert an outlier to illustrate robustness.
vY[50000] = 500

#Here, the observations are equidistant, but the code can handle unevenly spaced observations.
timestamps = seq(34200 , 57600 , length.out = iT)
testtimes = seq(34200, 57600, 60)
logprices = log(vY)

#calculating drift burst hypothesis

DBHtStat = drift_bursts(timestamps,  logprices,
                        testtimes, PreAverage = 5, AcLag = -1L,
                        Mean_bandwidth = 300L, Variance_bandwidth = 900L,
                        bParallelize = FALSE)

#plot series
plot(vY , x = timestamps/86400 , type = 'l')
#plot test statistic
plot(DBHtStat$DriftBursts, x = testtimes/86400, type = 'l')

## Not run:
################### same example with xts object:
#Set parameter values of the simulation
iT = 66500; dSigma = 0.3; dPhi = 0.98; dMu = -10;
#set seed for reproducibility
set.seed(123)
#Simulate the series
vY = 500+cumsum(StochasticVolatilitySim(iT, dSigma, dPhi, dMu))

#insert an outlier to illustrate robustness.
vY[50000] = 500

#Here, the observations are equidistant, but the code can handle unevenly spaced observations.
timestamps = seq(34200 , 57600 , length.out = iT)
StartTime = strptime("1970-01-01 00:00:00.0000", "
Tradetime = StartTime + timestamps
testtimes = seq(34200, 57600, 60)


price = xts(vY, Tradetime)


DBH = drift_bursts(time = NULL,  log(price),
```

```
                    testtimes, PreAverage = 5, AcLag = -1L,
                    Mean_bandwidth = 300L, Variance_bandwidth = 900L,
                    bParallelize = FALSE)

plot(DBH$DriftBursts)

#check for equality
all.equal(as.numeric(DBHtStat$DriftBursts), as.numeric(DBH$DriftBursts))

## End(Not run)
```

Index