Package 'pvar'

November 15, 2018

Version 2.2.5

Title Calculation and Application of p-Variation

Date 2018-11-12

Author Vygantas Butkus

Maintainer Vygantas Butkus <Vygantas.Butkus@gmail.com>

Description The calculation of p-variation of the finite sample data.
This package is a realisation of the procedure described in Butkus, V. & Norvaisa, R. Lith Math J (2018). <doi: 10.1007/s10986-018-9414-3>
The formal definitions and reference into literature are given in vignette.

LazyData yes

Imports Rcpp (>= 0.11.1)

LinkingTo Rcpp

Suggests e1071, testthat, knitr, formatR,

VignetteBuilder knitr

License GPL-2

RoxygenNote 6.1.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-11-15 08:20:02 UTC

R topics documented:

`pvar-package` *p-variation calculation and application*

Description

This package deals with p-variation for the sample (i.e. the sequence of data values). It gives opportunity to calculate the p-variation for the sample – this is the main purpose of this package. Nonetheless, it could be used to calculate p-variation for arbitrary piecewise monotonic function as well. Moreover, the package includes one example of practical application of the p-variation.

Details

Package:	pvar
Type:	Package
Version:	2.2.5
Date:	2016-05-17
License:	GPL-2
Institution:	Vilnius University Faculty of Mathematics and Informatics

This package is about p-variation. It deals with p-variation of a finite sample data values. To be precise, lets star with the definitions. Originally p-variation is defined for a functions.

For a function $f : [0, 1] \to R$ and $0 < p < \infty$ p-variation is defined as

$$v_p(f) = \sup \left\{ \sum_{i=1}^{m} |f(t_i) - f(t_{i-1})|^p : 0 = t_0 < t_1 < \ldots < t_m = 1, m \geq 1 \right\}$$

Analogically, for a sequences of values $X_0, X_1, ..., X_n$, the p-variation is defined as

$$v_p(\{X_i\}_{i=0}^n) = \max \left\{ \sum_{i=1}^{k} |X_{j_i} - X_{j_{i-1}}|^p : 0 = j_0 < j_1 < \ldots < j_k = n,\ k = 1, 2, ..., n \right\}$$

The points $0 = t_0 < t_1 < \ldots < t_m = 1$ (or $0 = j_0 < j_1 < \ldots < j_k = n$) that achieves the maximums is called a supreme partition (or just a partition for short).

There are two main functions that this package is all about, namely it is `pvar` and `PvarBreakTest`. The main function in this package is `pvar`. It calculates the p-variation and the partition. And the function `PvarBreakTest` is one of the examples of p-variation applications. It performs structural break test of vector `x` that exams whether there are multiple shifts in mean inside vector `x`.

All other functions are loaded only for supporting and illustrating purposes.

Author(s)

Author and Maintainer: Vygantas Butkus <Vygantas.Butkus@gmail.com>.

Special thanks to Rimas Norvaisa the supervisor of my studies.

References

[1] V. Butkus, R. Norvaisa. Lith Math J (2018). https://doi.org/10.1007/s10986-018-9414-3

[2] R. M. Dudley, R. Norvaisa. An Introduction to p-variation and Young Integrals, Cambridge, Mass., 1998.

[3] R. M. Dudley, R. Norvaisa. Differentiability of Six Operators on Nonsmooth Functions and p-Variation, Springer Berlin Heidelberg, Print ISBN 978-3-540-65975-4, Lecture Notes in Mathematics Vol. 1703, 1999.

[4] R. Norvaisa, A. Rackauskas. Convergence in law of partial sum processes in p-variation norm. Lth. Math. J., 2008., Vol. 48, No. 2, 212-227.

[5] J. Qian. The p-variation of Partial Sum Processes and the Empirical Process. The Annals of Probability, 1998, Vol. 26, No. 3, 1370-1383.

See Also

The main function is `pvar` - it finds p-variation and the partition that maximizes `Sum_p` function.

Other important functions is `PvarBreakTest` it performs structural break test of vector `x` by calculating p-variations of `BridgeT(x)` (see `BridgeT`).

`AddPvar` *Addition of p-variation*

Description

Merges two objects of p-variation and effectively recalculates the p-variation of joined sample.

Usage

```
AddPvar(PV1, PV2, AddIfPossible = TRUE)
```

Arguments

`PV1`	an object of the class `pvar`.
`PV2`	an object of the class `pvar`, which has the same `p` value as PV1 object.
`AddIfPossible`	`logical`. If TRUE (the default), then is is assumed, that two samples has common point. So, the end of PV1 and the begging of PV2 will be treated as one point if it has the same value.

Details

Note: a short form of `AddPvar(PV1, PV2` is `PV1 + PV2`.

Value

An object of the class pvar. See pvar.

Examples

```
### creating two pvar objects:
x = rwiener(1000)
PV1 = pvar(x[1:500], 2)
PV2 = pvar(x[500:1000], 2)

layout(matrix(c(1,3,2,3), 2, 2))
plot(PV1)
plot(PV2)
plot(AddPvar(PV1, PV2))
layout(1)

### AddPvar(PV1, PV2) is eqivavalent to PV1 + PV2
IsEqualPvar(AddPvar(PV1, PV2), PV1 + PV2)
```

BridgeT *Bridge transformation*

Description

Transforms data by Bridge transformation.

Usage

```
BridgeT(x, normalize = TRUE)
```

Arguments

x	x a numeric vector of data values.
normalize	logical, indicating whether the vector should be normalized.

Details

Let n denotes the length ox x. For each $m \in [1, n]$ bridge transformations BridgeT is defined as

$$BridgeT(m, x) = \left\{ \sum_{i=1}^{m} x_i - \frac{m}{n} \sum_{i=1}^{n} x_i \right\}.$$

Meanwhile, the transformation with normalization is

$$BridgeT(m, x) = \frac{1}{\sqrt{nvar(x)}} \left\{ \sum_{i=1}^{m} x_i - \frac{m}{n} \sum_{i=1}^{n} x_i \right\}.$$

Value

A numeric vector.

See Also

PvarBreakTest, rbridge

Examples

```
x <- rnorm(1000)
Bx <- BridgeT(x, FALSE)

op <- par(mfrow=c(2,1),mar=c(4,4,2,1))
plot(cumsum(x), type="l")
plot(Bx, type="l")
par(op)
```

ChangePoints *Change Points of a* numeric *vector*

Description

Finds changes points (i.e. corners) in the numeric vector.

Usage

```
ChangePoints(x)
```

Arguments

x	numeric vector.

Details

The end points of the vector will be always included in the results.

Value

The vector of index of change points.

Examples

```
x <- rwiener(100)
cid <- ChangePoints(x)
plot(x, type="l")
points(time(x)[cid], x[cid], cex=0.5, col=2, pch=19)
```

DataSets *Data sets of Monte-Carlo simulations results*

Description

The test PvarBreakTest uses quantiles from Monte-Carlo simulations. The results of the simulations are saved in these data sets.

Usage

```
PvarQuantileDF
MeanCoef
SdCoef
```

Format

the PvarQuantileDF is a data.frame with fields prob an Qaunt. The field brob represent the probability and Quant gives correspondingly quantile. MeanCoef and SdCoef is a named vector used in functions getMean and getSd.

Details

The distribution of p-variation of BridgeT(x) are unknown, therefore it was approximated form Monte-Carlo simulation based on 140 millions iterations. The data frame PvarQuantile summarize the distribution of normalized statistics. Meanwhile, MeanCoef and SdCoef defines the coefficients of functional form of mean and sd statistics of PvarBreakTest statistics (see getMean).

Author(s)

Vygantas Butkus <Vygantas.Butkus@gmail.com>

Source

Monte-Carlo simulation

IsEqualPvar *Test if two 'pvar' objects are equivalent.*

Description

Two pvar objects are considered to be equal if they have the same x, p, value and the same value of x in the points of partition (the index of partitions are not necessary the same). All other tributes like dname or TimeLabel are not important.

Usage

```
IsEqualPvar(pv1, pv2)
```

Arguments

pv1	an object of the class pvar.
pv2	an object of the class pvar.

Examples

```
x <- rwiener(100)
pv1 <- pvar(x, 2)
pv2 <- pvar(x[1:50], 2) + pvar(x[50:101], 2)
IsEqualPvar(pv1, pv2)
```

---

pvar *p-variation calculation*

Description

Calculates p-variation of the sample.

Usage

```
pvar(x, p, TimeLabel = as.vector(time(x)), LSI = 3)

## S3 method for class 'pvar'
summary(object, ...)

## S3 method for class 'pvar'
plot(x, main = "p-variation", ylab = x$dname,
  sub = "p=" %.% round(x$p, 5) %.% ", p-variation: " %.%
  formatC(x$value, 5, format = "f"), col.PP = 2, cex.PP = 0.5, ...)
```

Arguments

x	a (non-empty) numeric vector of data values or an object of the class pvar.
p	a positive number indicating the power p in p-variation.
TimeLabel	numeric, a time index of x. Used only for plotting.
LSI	a length of small interval. It must be a positive odd number. This parameter do not have effect on final result, but might influence the speed of calculation.
object	an objct of the class pvar.
...	further arguments.
main	a main parameter in plot function.
ylab	a ylab parameter in plot function.

sub	a sub parameter in plot function.
col.PP	the color of partition points.
cex.PP	the cex of partition points.

Details

This function is the main function in this package. It calculates the p-variation of the sample. The formal definition is given in pvar-package.

Value

An object of the class pvar. Namely, it is a list that contains

value	a value of p-variation.
x	a vector of original data x.
p	the value of p.
partition	a vector of indexes that indicates the partition that achieves the maximum.
dname	a name of data vector (optional).
TimeLabel	a time label of x (optional).

Author(s)

Vygantas Butkus <Vygantas.Butkus@gmail.com>

See Also

IsEqualPvar, AddPvar, PvarBreakTest.

Examples

```
### randomised data:
x = rbridge(1000)

### the main functions:
pv = pvar(x, 2)
print(pv)
summary(pv)
plot(pv)

### The value of p-variation is
pv; Sum_p(x[pv$partition], 2)

### The meaning of supreme partition points:
pv.PP = pvar(x[pv$partition], TimeLabel=time(x)[pv$partition], 2)
pv.PP == pv.PP
op <- par(mfrow = c(2, 1), mar=c(2, 4, 4, 1))
plot(pv, main='pvar with original data')
plot(pv.PP, main='The same pvar without redundant points')
par(op)
```

PvarBreakTest *Structural break test*

Description

This function performs structural break test that is based on p-variation.

Usage

```
PvarBreakTest(x, TimeLabel = as.vector(time(x)), alpha = 0.05,
  FullInfo = TRUE)

## S3 method for class 'PvarBreakTest'
plot(x, main1 = "Data",
  main2 = "Bridge transformation", ylab1 = x$dname,
  ylab2 = "BridgeT(" %.% x$dname %.% ")", sub2 = NULL,
  col.PP = 3, cex.PP = 0.5, col.BP = 2, cex.BP = 1, cex.DP = 0.5,
  ...)

## S3 method for class 'PvarBreakTest'
summary(object, ...)
```

Arguments

x	a numeric vector of data values or an object of class pvar.
TimeLabel	numeric, a time index of x. Used only for plotting.
alpha	a small number greater then 0. It indicates the significant level of the test.
FullInfo	logical. If TRUE (the default) the function will return an object of the class PvarBreakTest that saves all useful information. Otherwise only the statistics will by returned.
main1	the main parameter of the data graph.
main2	the main parameter of the Bridge transformation graph.
ylab1	the ylab parameter of the data graph.
ylab2	the ylab parameter of the Bridge transformation graph.
sub2	the sub parameter of the Bridge transformation graph. By default it reports the number of break points.
col.PP	the color of partition points.
cex.PP	the cex of partition points.
col.BP	the color of break points.
cex.BP	the cex of break points.
cex.DP	the cex of data points.
...	further arguments, passed to print.
object	the object of the class PvarBreakTest.

Details

Lets x be a data that should be tested of structural breaks. Then the p-variation of the BridgeT(x) with p=4 is the test's statistics.

The quantiles of H0 distribution is based on Monte-Carlo simulation of 140 millions iterations. The test is reliable then length(x) is between 100 and 10000. The test might work with other lengths too, but it is not tested well. The test will not compute then length(x)<20.

Value

If FullInfo=TRUE then function returns an object of the class PvarBreakTest. It is the list that contains:

Stat	a value of statistics (p-variation of transformed data).
CriticalValue	the critical value of the test according to significant level.
alpha	the significant level.
p.value	approximate p-value.
reject	logical. If TRUE, the H0 was rejected.
dname	the name of data vector.
p	the power in p-variation calculus. The test performs only with the p=4.
x	a vector of original data.
y	a vector of transformed data (y=BridgeT(x)).
Timelabel	time label of x. Used only for ploting.
BreakPoints	the indexes of break points suggestion.
Partition	a vector of indexes that indicates the partition of y that achieves the p-variation maximum.

Author(s)

Vygantas Butkus <Vygantas.Butkus@gmail.com>

References

The test was proposed by A. Rackaskas. The test is based on the results given in the flowing article

[1] R. Norvaisa, A. Rackauskas. Convergence in law of partial sum processes in p-variation norm. Lth. Math. J., 2008., Vol. 48, No. 2, 212-227.

See Also

Tests statistics is pvar of the data BridgeT(x)(see BridgeT) with (p=4). The critical value and the approximate p-value of the test might by found by functions PvarQuantile and PvarPvalue.

Examples

```
set.seed(1)
MiuDiff <- 0.3
x <- rnorm(250*4, rep(c(0, MiuDiff, 0, MiuDiff), each=250))

plot(x, pch=19, cex=0.5, main='original data, with several shifts of mean')
k <- 50
moveAvg <- filter(x, rep(1/k, k))
lines(time(x), moveAvg, lwd=2, col=2)
legend('topleft', c('sample', 'moving average (k='%.%k%.%')'),
       lty=c(NA,1), lwd=c(NA, 2), col=1:2, pch=c(19,NA), pt.cex=c(0.7,1)
       ,inset = .03, bg='antiquewhite1')

xtest <- PvarBreakTest(x)
plot(xtest)
```

`PvarQuantile` *Quantiles and probabilities of p-variation*

Description

The distribution of p-variation of `BridgeT(x)` depends on `n=length(x)`. This fact is important for getting appropriate quantiles (or p-value). These functions helps to deal with it.

Usage

```
PvarQuantile(n, prob = c(0.9, 0.95, 0.99), DF = PvarQuantileDF)

PvarPvalue(n, stat, DF = PvarQuantileDF)

getMean(n, bMean = MeanCoef)

getSd(n, bSd = SdCoef)

NormalisePvar(x, n, bMean = MeanCoef, bSd = SdCoef)
```

Arguments

`n`	a positive integer indicating the length of data vector.
`prob`	cumulative probabilities of p-variation distribution.
`DF`	a `data.frame` that links `prob` and `stat` .
`stat`	a vector of p-variation statistics.
`bMean`	a coefficient vector that defines a function of the mean of p-variation.
`bSd`	a coefficient vector that defines a function of the standard deviation of p-variation.
`x`	a numeric vector of data values.

Details

The distribution of p-variance is form Monte-Carlo simulation based on 140 millions iterations. The data frame PvarQuantileDF saves the results of Monte-Carlo simulation.

Meanwhile, MeanCoef and SdCoef defines the coefficients of functional form (conditional on n) of mean and sd statistics.

A functional form of mean and sd statistics are the same, namely

$$f(n) = b_1 + b_2 n_2^b.$$

The coefficients (b_1, b_2, b_3) are saved in vectors MeanCoef and SdCoef. Those vectors are estimated with nls function form Monte-Carlo simulation.

Value

Functions PvarQuantile and PvarPvalue returns a corresponding value quantile or the probability. Functions getMean and getSd returns a corresponding value of mean and sd statistics. Function NormalisePvar returns normalize values.

Note

Arguments n, stat and prob might be vectors, but they can't be vectors simultaneously (at least one of then must be a number).

See Also

PvarBreakTest, PvarQuantileDF, NormalisePvar, getMean, getSd

---

rwiener *Random process generators*

Description

Generate a trajectory of random processes.

Usage

```
rwiener(frequency = 1000, end = 1)

rbridge(frequency = 1000, end = 1)

rcumbin(frequency = 1000, end = 1)
```

Arguments

frequency	a number specifying the size of trajectory vector. The trajectory will start at point 0 and will have frequency more observations. The length of the results will be frequency+1 .
end	a number. The end point of the process in the 'time' scale.

Details

rwiener generate Wiener process via partial sums process and rbridge generate Brownian bridge via rwiener. The original code of rwiener and rbridge was written in the package e1071. In this package these functions was modified to include leading zero in the beginning of the sample.

rcumbin generate partial sums process from random variables with values -1, 0, 1.

Value

A time series containing a simulated realization of random processes. The length of time series is frequency+1, since zero is always included in the beginning of the sample.

Sum_p *p-variation summation function*

Description

It is the sum of absolute differences in the power of p.

Usage

```
Sum_p(x, p, lag = 1)
```

Arguments

x	a numeric vector of data values.
p	a number indicating the power in summing function.
lag	a number, indicating the lag of differences.

Details

This is a function that must be maximized by taking a proper subset of x, i.e. if prt is a p-variation partition of sample x, then Sum_p(x[prt], p) == pvar(x, p)$value.

Value

The number equal to sum((abs(diff(x, lag)))^p)

See Also

pvar

Examples

```
x = rbridge(1000)
pv = pvar(x, 2); pv
# Sum_p in supreme partition and the value form pvar must match
Sum_p(x[pv$partition], 2)
pv
```

`%.%` *Concatenate strings*

Description

Concatenate Strings

Usage

```
x %.% y
```

Arguments

x	asd
y	asd

Details

The same result may be achieved with `paste`, but in some circumstance this function is more user friendly.

Value

A character string of the concatenated values.

See Also

`paste`

Examples

```
paste('I ', 'love ', 'R.', sep='')
'I ' %.% 'love ' %.% 'R.'

x = c(2,1,6,7,9)
paste('The length of vector (', paste(x , sep='', collapse =','), ') is ', length(x) , sep='')
'The length of vector (' %.% paste(x , sep='', collapse =',') %.% ') is ' %.% length(x)
```

Index