Package ‘changepoint’

November 3, 2022

Type Package

Title Methods for Changepoint Detection

Version 2.2.4

Date 2022-10-31

Maintainer Rebecca Killick <r.killick@lancs.ac.uk>

BugReports https://github.com/rkillick/changepoint/issues

URL https://github.com/rkillick/changepoint/

Description
Implements various mainstream and specialised changepoint methods for finding single and multiple changepoints within data. Many popular non-parametric and frequentist methods are included. The cpt.mean(), cpt.var(), cpt.meanvar() functions should be your first point of call.

Depends R(>= 3.1), methods, stats, zoo(>= 0.9-1)

Suggests testthat, vdiffr

License GPL

LazyData true

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-11-03 22:30:06 UTC

Author Rebecca Killick [aut, cre],
      Kaylea Haynes [aut],
      Idris Eckley [ths],
      Paul Fearnhead [ctb, ths],
      Robin Long [ctb],
      Jamie Lee [ctr]

R topics documented:

changepoint-package ........................................ 2
BINSEG ....................................................... 3
class_input ................................................ 5
Description

Implements various mainstream and specialised changepoint methods for finding single and multiple changepoints within data. Many popular non-parametric and frequentist methods are included. Users should start by looking at the documentation for `cpt.mean()`, `cpt.var()` and `cpt.meanvar()`.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>changepoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>2.2.4</td>
</tr>
<tr>
<td>Date</td>
<td>2022-10-31</td>
</tr>
<tr>
<td>License</td>
<td>GPL</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
</tbody>
</table>

Author(s)

Rebecca Killick <r.killick@lancs.ac.uk>, Kaylea Haynes <k.haynes1@lancs.ac.uk> with contributions from Idris A. Eckley <i.eckley@lancs.ac.uk>, Paul Fearnhead <p.fearnhead@lancs.ac.uk>.

Maintainer: Rebecca Killick <r.killick@lancs.ac.uk>

References


See Also

cpt.mean, cpt.var, cpt.meanvar

Examples

```r
# change in variance
set.seed(1)
x=c(rnorm(100,0,1),rnorm(100,0,10))
ansvar=cpt.var(x)
plot(ansvar)
print(ansvar) # identifies 1 changepoint at 100

# change in mean
y=c(rnorm(100,0,1),rnorm(100,5,1))
ansmean=cpt.mean(y)
plot(ansmean,cpt.col='blue')
print(ansmean)

# change in mean and variance
z=c(rnorm(100,0,1),rnorm(100,2,10))
ansmeanvar=cpt.meanvar(z)
plot(ansmeanvar,cpt.width=3)
print(ansmeanvar)
```

---

**BINSEG**

Binary Segmentation - Only intended for developer use.

Description

Implements the Binary Segmentation method for identifying changepoints in a given set of summary statistics for a specified cost function and penalty.

This function is called by `cpt.mean`, `cpt.var` and `cpt.meanvar` when `method="BinSeg"`. This is not intended for use by regular users of the package. It is exported for developers to call directly for speed increases or to fit alternative cost functions.

WARNING: No checks on arguments are performed!

Usage

`BINSEG(sumstat, pen = 0, cost_func = "norm.mean", shape = 1, minseglen = 2, Q=5)`
Arguments

- **sumstat**: A matrix containing the summary statistics of data within which you wish to find a changepoint. Currently assumes 3 columns and uses the number of rows as the length of the data +1 (initial value of 0).
- **pen**: Default choice is 0, this should be evaluated elsewhere and a numerical value entered. This should be positive - this isn’t checked but results are meaningless if it isn’t.
- **cost_func**: The friendly name of the cost function to be called in C. If using your own cost function, this must be the name of the C function to use.
- **shape**: Only required for cost_func="Gamma", default is 1. Must be a positive value, this isn’t checked.
- **minseglen**: Positive integer giving the minimum segment length (no. of observations between changes), default is 2. No checks are performed on the input value so it could be larger than feasible to have changes in the data.
- **Q**: The maximum number of changepoints to search for (positive integer). No checks are performed and so a number larger than allowed can be input.

Details

This function is used as a wrapper function to implement the Binary Segmentation algorithm in C. It simply creates the necessary worker vectors, ensures all inputs are the correct type, and passes everything to the C function.

This function is exported for developer use only. It does not perform any checks on inputs (other than type coercion) and is simply a wrapper function for the C code.

Value

A list is returned with elements:

- **cps**: 2xQ Matrix containing the changepoint positions on the first row and the test statistic on the second row in the order identified.
- **cpts**: Ordered list of optimal number of changepoints ending with n.
- **op.cpts**: The optimal number changepoint locations for the penalty supplied.
- **pen**: Penalty used to find the optimal number of changepoints.

Author(s)

Rebecca Killick

References


See Also

cpt.mean.cpt.meanvar.plot-methods.cpt
**class_input**

Input all required arguments into cpt classes - Only intended for developer use.

**Description**

This function helps to input all the necessary information into the correct format for cpt and cpt.range classes.

This function is called by cpt.mean, cpt.var and cpt.meanvar when class=TRUE. This is not intended for use by regular users of the package. It is exported for developers to call directly for speed and convenience.

WARNING: No checks on arguments are performed!

**Usage**

```r
class_input(data, cpttype, method, test.stat, penalty, pen.value, minseglen, param.estimates, out=list(), Q=NA, shape=NA)
```

**Arguments**

- **data**: Data used in changepoint analysis, see `cpt.mean` for further details.
- **cpttype**: Type of changepoint analysis performed as a text string, e.g. "Mean", "Mean and Variance".
- **method**: Method used as a text string, see `cpt.mean` for further details.
- **test.stat**: The assumed test statistic / distribution of the data as a text string, see `cpt.mean`, `cpt.meanvar` or `cpt.var` for further details.
- **penalty**: Penalty used as a text string, see `cpt.mean` for further details.
- **pen.value**: Numerical penalty value used in the analysis (positive).
- **minseglen**: Minimum segment length used in the analysis (positive integer).
- **param.estimates**: Logical. If TRUE then parameter estimates are calculated. If FALSE no parameter estimates are calculated and the slot is blank in the returned object.
- **out**: List of output from `BINSEG`, `PELT` or other method used. Function assumes that method and format of out match.
- **Q**: The value of Q used in the BinSeg or SegNeigh methods.
- **shape**: Value of the assumed known shape parameter required when test.stat="Gamma".
Details
This function takes all the input required for the cpt or cpt.range classes and enters it into the object.
This function is exported for developer use only. It does not perform any checks on inputs and is simply a convenience function for converting the output of the worker functions into a nice format for the cpt and cpt.range classes.

Value
An object of class cpt or cpt.range as appropriate filled with the given attributes.

Author(s)
Rebecca Killick

See Also
cpt.var, cpt.mean, plot-methods.cpt

Examples
#This function should only be used by developers, see its use in cpt.mean, cpt.var and cpt.meanvar.

cpt.mean(data, penalty="MBIC", pen.value=0, method="AMOC", Q=5, test.stat="Normal", class=TRUE, param.estimates=TRUE, minseglen=1)

Arguments
data A vector, ts object or matrix containing the data within which you wish to find a changepoint. If data is a matrix, each row is considered a separate dataset.
penalty Choice of "None", "SIC", "BIC", "MBIC", AIC", "Hannan-Quinn", "Asymptotic", "Manual" and "CROPS" penalties. If Manual is specified, the manual penalty is contained in the pen.value parameter. If Asymptotic is specified, the theoretical type I error is contained in the pen.value parameter. If CROPS is specified, the penalty range is contained in the pen.value parameter; note this is a vector of length 2 which contains the minimum and maximum penalty value.
Note CROPS can only be used if the method is "PELT". The predefined penalties listed DO count the changepoint as a parameter, postfix a 0 e.g. "SIC0" to NOT count the changepoint as a parameter.

**pen.value**
The theoretical type I error e.g. 0.05 when using the Asymptotic penalty. A vector of length 2 (min, max) if using the CROPS penalty. The value of the penalty when using the Manual penalty option - this can be a numeric value or text giving the formula to use. Available variables are, n=length of original data, null=null likelihood, alt=alternative likelihood, tau=proposed changepoint, diffparam=difference in number of alternative and null parameters.

**method**
Choice of "AMOC", "PELT", "SegNeigh" or "BinSeg".

**Q**
The maximum number of changepoints to search for using the "BinSeg" method. The maximum number of segments (number of changepoints + 1) to search for using the "SegNeigh" method.

**test.stat**
The assumed test statistic / distribution of the data. Currently only "Normal" and "CUSUM" supported.

**class**
Logical. If TRUE then an object of class cpt is returned.

**param.estimates**
Logical. If TRUE and class=TRUE then parameter estimates are returned. If FALSE or class=FALSE no parameter estimates are returned.

**minseglen**
Positive integer giving the minimum segment length (no. of observations between changes), default is the minimum allowed by theory.

**Details**
This function is used to find changes in mean for data using the test statistic specified in the test.stat parameter. The changes are found using the method supplied which can be single changepoint (AMOC) or multiple changepoints using exact (PELT or SegNeigh) or approximate (BinSeg) methods. A changepoint is denoted as the last observation of the segment / regime.

**Value**
If class=TRUE then an object of S4 class "cpt" is returned. The slot cpts contains the changepoints that are returned. For class=FALSE the structure is as follows.

If data is a vector (single dataset) then a vector/list is returned depending on the value of method. If data is a matrix (multiple datasets) then a list is returned where each element in the list is either a vector or list depending on the value of method.

If method is AMOC then a vector (one dataset) or matrix (multiple datasets) is returned, the columns are:

- **cpt**
  The most probable location of a changepoint if a change was identified or NA if no changepoint.
- **p value**
  The p-value of the identified changepoint.

If method is PELT then a vector is returned containing the changepoint locations for the penalty supplied. This always ends with n. If the penalty is CROPS then a list is returned with elements:
cpt.out A data frame containing the value of the penalty value where the number of segmentations changes, the number of segmentations and the value of the cost at that penalty value.

changepoints The optimal changepoint for the different penalty values starting with the lowest penalty value.

If method is SegNeigh then a list is returned with elements:

cps Matrix containing the changepoint positions for 1,...,Q changepoints.
op.cpts The optimal changepoint locations for the penalty supplied.
pen Penalty used to find the optimal number of changepoints.
like Value of the -2*log(likelihood ratio) + penalty for the optimal number of changepoints selected.

If method is BinSeg then a list is returned with elements:

cps 2xQ Matrix containing the changepoint positions on the first row and the test statistic on the second row.
op.cpts The optimal changepoint locations for the penalty supplied.
pen Penalty used to find the optimal number of changepoints.

Author(s)
Rebecca Killick

References


See Also

cpt.var, cpt.meanvar, plot-methods.cpt
Examples

# Example of a change in mean at 100 in simulated normal data
set.seed(1)
x=c(rnorm(100,0,1),rnorm(100,10,1))
cpt.mean(x,penalty="SIC",method="AMOC",class=FALSE) # returns 100 to show that the null hypothesis
# was rejected and the change in mean is at 100 and the confidence level is 1.
ans=cpt.mean(x,penalty="Asymptotic",pen.value=0.01,method="AMOC")
cpts(ans)# returns 100 to show that the null hypothesis was rejected, the change in mean is at 100
# and we are 99% confident of this result
cpt.mean(x,penalty="Manual",pen.value=0.8,method="AMOC",test.stat="CUSUM")
# returns 101 as the changepoint location

# Example of multiple changes in mean at 50,100,150 in simulated normal data
set.seed(1)
x=c(rnorm(50,0,1),rnorm(50,5,1),rnorm(50,10,1),rnorm(50,3,1))
cpt.mean(x,penalty="Manual",pen.value="2*log(n)",method="BinSeg",Q=5,class=FALSE)
# returns optimal number of changepoints is 3, locations are 50,100,150.

# Example of using the CROPS penalty in data set above
set.seed(1)
x=c(rnorm(50,0,1),rnorm(50,5,1),rnorm(50,10,1),rnorm(50,3,1))
out=cpt.mean(x, pen.value = c(4,1500),penalty = "CROPS",method = "PELT")
cpts.full(out) # returns 7 segmentations for penalty values between 4 and 1500.
# We find segmentations with 7, 5, 4, 3, 2, 1 and 0 changepoints.
# Note that the empty final row indicates no changepoints.
pen.value.full(out) # gives associated penalty transition points
# CROPS does not give an optimal set of changepoints thus we may wish to explore further
plot(out,diagnostic=TRUE)
# looks like the segmentation with 3 changepoints, 50,100,150 is the most appropriate
plot(out,ncpts=3)

# Example multiple datasets where the first row has multiple changes in mean and the second row has
# no change in mean
set.seed(1)
x=c(rnorm(50,0,1),rnorm(50,5,1),rnorm(50,10,1),rnorm(50,3,1))
y=rnorm(200,0,1)
z=rbind(x,y)
cpt.mean(z,penalty="Asymptotic",pen.value=0.01,method="SegNeigh",Q=5,class=FALSE) # returns list
# that has two elements, the first has 3 changes in mean and variance at 50,100,150 and the second
# has no changes in variance
ans=cpt.mean(z,penalty="Asymptotic",pen.value=0.01,method="PELT")
cpts(ans[[1]]) # same results as for the SegNeigh method.
cpts(ans[[2]]) # same results as for the SegNeigh method.
Description

Calculates the optimal positioning and (potentially) number of changepoints for data using the user specified method.

Usage

cpt.meanvar(data,penalty="MBIC",pen.value=0,method="AMOC",Q=5,test.stat="Normal",class=TRUE,param.estimates=TRUE,shape=1,minseglen=2)

Arguments

data A vector, ts object or matrix containing the data within which you wish to find a changepoint. If data is a matrix, each row is considered a separate dataset.

penalty Choice of "None", "SIC", "BIC", "MBIC", "AIC", "Hannan-Quinn", "Asymptotic", "Manual" and "CROPS" penalties. If Manual is specified, the manual penalty is contained in the pen.value parameter. If Asymptotic is specified, the theoretical type I error is contained in the pen.value parameter. If CROPS is specified, the penalty range is contained in the pen.value parameter; note this is a vector of length 2 which contains the minimum and maximum penalty value. Note CROPS can only be used if the method is "PELT". The predefined penalties listed DO count the changepoint as a parameter, postfix a 0 e.g."SIC0" to NOT count the changepoint as a parameter.

pen.value The theoretical type I error e.g.0.05 when using the Asymptotic penalty. A vector of length 2 (min,max) if using the CROPS penalty. The value of the penalty when using the Manual penalty option - this can be a numeric value or text giving the formula to use. Available variables are, n=length of original data, null=null likelihood, alt=alternative likelihood, tau=proposed changepoint, diffparam=difference in number of alternative and null parameters.

method Choice of "AMOC", "PELT", "SegNeigh" or "BinSeg".

Q The maximum number of changepoints to search for using the "BinSeg" method. The maximum number of segments (number of changepoints + 1) to search for using the "SegNeigh" method.

test.stat The assumed test statistic / distribution of the data. Currently only "Normal", "Gamma", "Exponential" and "Poisson" are supported.

class Logical. If TRUE then an object of class cpt is returned.

param.estimates Logical. If TRUE and class=TRUE then parameter estimates are returned. If FALSE or class=FALSE no parameter estimates are returned.

shape Value of the assumed known shape parameter required when test.stat="Gamma".

minseglen Positive integer giving the minimum segment length (no. of observations between changes), default is the minimum allowed by theory.

Details

This function is used to find changes in mean and variance for data using the test statistic specified in the test.stat parameter. The changes are found using the method supplied which can be single
changepoint (AMOC) or multiple changepoints using exact (PELT or SegNeigh) or approximate (BinSeg) methods. A changepoint is denoted as the last observation of the segment / regime.

Value

If class=TRUE then an object of S4 class "cpt" is returned. The slot cpts contains the changepoints that are returned. For class=FALSE the structure is as follows.

If data is a vector (single dataset) then a vector/list is returned depending on the value of method. If data is a matrix (multiple datasets) then a list is returned where each element in the list is either a vector or list depending on the value of method.

If method is AMOC then a vector (one dataset) or matrix (multiple datasets) is returned, the columns are:

- cpt: The most probable location of a changepoint if a change was identified or NA if no changepoint.
- p value: The p-value of the identified changepoint.

If method is PELT then a vector is returned containing the changepoint locations for the penalty supplied. This always ends with n. If the penalty is CROPS then a list is returned with elements:

- cpt.out: A data frame containing the value of the penalty value where the number of segmentations changes, the number of segmentations and the value of the cost at that penalty value.
- changepoints: The optimal changepoints for the different penalty values starting with the lowest penalty value.

If method is SegNeigh then a list is returned with elements:

- cps: Matrix containing the changepoint positions for 1,...,Q changepoints.
- op.cpts: The optimal changepoint locations for the penalty supplied.
- pen: Penalty used to find the optimal number of changepoints.
- like: Value of the -2*log(likelihood ratio) + penalty for the optimal number of changepoints selected.

If method is BinSeg then a list is returned with elements:

- cps: 2xQ Matrix containing the changepoint positions on the first row and the test statistic on the second row.
- op.cpts: The optimal changepoint locations for the penalty supplied.
- pen: Penalty used to find the optimal number of changepoints.

Author(s)

Rebecca Killick
References


See Also
cpt.var, cpt.mean, plot-methods, cpt

Examples

```r
# Example of a change in scale parameter (mean and variance) at 100 in simulated gamma data
set.seed(1)
x=c(rgamma(100,shape=1,rate=1),rgamma(100,shape=1,rate=5))
cpt.meanvar(x,penalty="SIC",method="AMOC",test.stat="Gamma",class=FALSE,shape=1) # returns 97 to #show that the null hypothesis was rejected and the change in scale parameter is at 97
ans=cpt.meanvar(x,penalty="AIC",method="AMOC",test.stat="Gamma",shape=1)
cpts(ans) # returns 97 to show that the null hypothesis was rejected, the change in scale parameter is at 97

# Example of multiple changes in mean and variance at 50,100,150 in simulated normal data
set.seed(1)
x=c(rnorm(50,0,1),rnorm(50,5,3),rnorm(50,10,1),rnorm(50,3,10))
cpt.meanvar(x,penalty="Manual",pen.value="4*log(n)",method="BinSeg",Q=5,class=FALSE) # returns optimal number of changepoints is 4, locations are 50,100,150,152.

# Example of using the CROPS penalty in the above example
set.seed(1)
x=c(rnorm(50,0,1),rnorm(50,5,3),rnorm(50,10,1),rnorm(50,3,10))
out=cpt.meanvar(x,pen.value=c(2*log(length(x)),100*log(length(x))),penalty="CROPS",method="PELT")
cpts.full(out) # returns 6 segmentations for penalty values between 2log(n) and 100log(n).
```
We find segmentations with 9, 7, 4, 3, 1 and 0 changepoints. Note that the empty final row indicates no changepoints. pen.value.full(out) # gives associated penalty transition points CROPS does not give an optimal set of changepoints thus we may wish to explore further.

Example multiple datasets where the first row has multiple changes in mean and variance and the second row has no change in mean or variance.

set.seed(1)
x = c(rnorm(50, 0, 1), rnorm(50, 5, 3), rnorm(50, 10, 1), rnorm(50, 3, 10))
y = rnorm(200, 0, 1)
z = rbind(x, y)
cpt.meanvar(z, penalty = "Asymptotic", pen.value = 0.01, method = "SegNeigh", Q = 5, class = FALSE) # returns list that has two elements, the first has 3 changes in mean and variance at 50, 100, 150 and the second has no changes in mean or variance.

ans = cpt.meanvar(z, penalty = "Asymptotic", pen.value = 0.01, method = "PELT")
cpts(ans[[1]]) # same results as for the SegNeigh method.
cpts(ans[[2]]) # same results as for the SegNeigh method.

---

cpt.var  Identifying Changes in Variance

**Description**

Calculates the optimal positioning and (potentially) number of changepoints for data using the user specified method.

**Usage**

```r
cpt.var(data, penalty = "MBIC", pen.value = 0, know.mean = FALSE, mu = NA, method = "AMOC", Q = 5, test.stat = "Normal", class = TRUE, param.estimates = TRUE, minseglen = 2)
```

**Arguments**

- **data**: A vector, ts object or matrix containing the data within which you wish to find a changepoint. If data is a matrix, each row is considered a separate dataset.
- **penalty**: Choice of "None", "SIC", "BIC", "MBIC", "AIC", "Hannan-Quinn", "Asymptotic", "Manual" and "CROPS" penalties. If Manual is specified, the manual penalty is contained in the pen.value parameter. If Asymptotic is specified, the theoretical type I error is contained in the pen.value parameter. If CROPS is specified, the penalty range is contained in the pen.value parameter; note this is a vector of length 2 which contains the minimum and maximum penalty value. Note CROPS can only be used if the method is "PELT". The predefined penalties listed DO count the changepoint as a parameter, postfix a 0 e.g. "SIC0" to NOT count the changepoint as a parameter.
The theoretical type I error e.g. 0.05 when using the Asymptotic penalty. A vector of length 2 (min,max) if using the CROPS penalty. The value of the penalty when using the Manual penalty option - this can be a numeric value or text giving the formula to use. Available variables are, n=length of original data, null=null likelihood, alt=alternative likelihood, tau=proposed changepoint, diffparam=difference in number of alternative and null parameters.

Only required for test.stat="Normal". Logical, if TRUE then the mean is assumed known and mu is taken as its value. If FALSE, and mu=NA (default value) then the mean is estimated via maximum likelihood. If FALSE and the value of mu is supplied, mu is not estimated but is counted as an estimated parameter for decisions.

Only required for test.stat="Normal". Numerical value of the true mean of the data. Either single value or vector of length nrow(data). If data is a matrix and mu is a single value, the same mean is used for each row.

Choice of "AMOC", "PELT", "SegNeigh" or "BinSeg".

The maximum number of changepoints to search for using the "BinSeg" method. The maximum number of segments (number of changepoints + 1) to search for using the "SegNeigh" method.

The assumed test statistic / distribution of the data. Currently only "Normal" and "CSS" supported.

Logical. If TRUE then an object of class cpt is returned.

Logical. If TRUE and class=TRUE then parameter estimates are returned. If FALSE or class=FALSE no parameter estimates are returned.

Positive integer giving the minimum segment length (no. of observations between changes), default is the minimum allowed by theory.

This function is used to find changes in variance for data using the test statistic specified in the test.stat parameter. The changes are found using the method supplied which can be single changepoint (AMOC) or multiple changepoints using exact (PELT or SegNeigh) or approximate (BinSeg) methods. A changepoint is denoted as the last observation of the segment / regime. Note that for the test.stat="CSS" option the preset penalties are log(.) to allow comparison with test.stat="Normal".

If class=TRUE then an object of S4 class "cpt" is returned. The slot cpts contains the changepoints that are returned. For class=FALSE the structure is as follows.

If data is a vector (single dataset) then a vector/list is returned depending on the value of method. If data is a matrix (multiple datasets) then a list is returned where each element in the list is either a vector or list depending on the value of method.

If method is AMOC then a vector (one dataset) or matrix (multiple datasets) is returned, the columns are:

The most probable location of a changepoint if a change was identified or NA if no changepoint.
 cpt.var

p value  The p-value of the identified changepoint.

If method is PELT then a vector is returned containing the changepoint locations for the penalty supplied. This always ends with n. If the penalty is CROPS then a list is returned with elements:

cpt.out  A data frame containing the value of the penalty value where the number of segmentations changes, the number of segmentations and the value of the cost at that penalty value.

segmentations  The optimal segmentations for the different penalty values starting with the lowest penalty value.

If method is SegNeigh then a list is returned with elements:

cps  Matrix containing the changepoint positions for 1,...,Q changepoints.

op.cpts  The optimal changepoint locations for the penalty supplied.

pen  Penalty used to find the optimal number of changepoints.

like  Value of the -2*log(likelihood ratio) + penalty for the optimal number of changepoints selected.

If method is BinSeg then a list is returned with elements:

cps  2xQ Matrix containing the changepoint positions on the first row and the test statistic on the second row.

op.cpts  The optimal changepoint locations for the penalty supplied.

pen  Penalty used to find the optimal number of changepoints.

Author(s)

Rebecca Killick

References


See Also

cpt.mean, cpt.meanvar, plot-methods, cpt

Examples

# Example of a change in variance at 100 in simulated normal data
set.seed(1)
x <- c(rnorm(100, 0, 1), rnorm(100, 0, 10))
cpt.var(x, penalty = "SIC", method = "AMOC", class = FALSE) # returns 100 to show that the null hypothesis was rejected and the change in variance is at 100
ans <- cpt.var(x, penalty = "Asymptotic", pen.value = 0.01, method = "AMOC")
cpts(ans) # returns 100 to show that the null hypothesis was rejected, the change in variance is at #100 and we are 99% confident of this result

# Example of multiple changes in variance at 50, 100, 150 in simulated data
set.seed(1)
x <- c(rnorm(50, 0, 1), rnorm(50, 0, 10), rnorm(50, 0, 5), rnorm(50, 0, 1))
cpt.var(x, penalty = "Manual", pen.value = "log(2+log(n))", method = "BinSeg", test.stat = "CSS", Q = 5, class = FALSE) # returns optimal number of changepoints is 4, locations are 50, 53, 99, 150.

# Example of using CROPS in the above example
set.seed(1)
x <- c(rnorm(50, 0, 1), rnorm(50, 0, 10), rnorm(50, 0, 5), rnorm(50, 0, 1))
out <- cpt.var(x, pen.value = c(log(length(x)), 100 * log(length(x))), penalty = "CROPS", method = "PELT")
cpts.full(out) # returns 7 segmentations for penalty values between log(n) and 100 log(n).
# We find segmentations with 7, 5, 4, 3, 2, 1 and 0 changepoints.
# Note that the empty final row indicates no changepoints.
pen.value.full(out) # gives associated penalty transition points
# CROPS does not give an optimal set of changepoints thus we may wish to explore further
plot(out, diagnostic = TRUE)
# looks like the segmentation with 3 changepoints, 50, 100, 150 is the most appropriate
plot(out, ncpts = 3)

# Example multiple datasets where the first row has multiple changes in variance and the second row has no change in variance
set.seed(10)
x <- c(rnorm(50, 0, 1), rnorm(50, 0, 10), rnorm(50, 0, 5), rnorm(50, 0, 1))
y <- rnorm(200, 0, 1)
z <- rbind(x, y)
cpt.var(z, penalty = "Asymptotic", pen.value = 0.01, method = "SegNeigh", Q = 5, class = FALSE) # returns list that has two elements, the first has 3 changes in variance at 50, 100, 149 and the second has no changes in variance
ans <- cpt.var(z, penalty = "Asymptotic", pen.value = 0.01, method = "PELT")
cpts(ans[[1]]) # same results as for the SegNeigh method.
cpts(ans[[2]]) # same results as for the SegNeigh method.
**Description**

Uses the function parameters to decide if a proposed changepoint is a true changepoint or due to random variability. Test is conducted using the user specified penalty.

This function is called by `cpt.mean`, `cpt.var` and `cpt.meanvar` when `method="AMOC"`. This is not intended for use by regular users of the package. It is exported for developers to call directly for speed increases or to fit alternative cost functions.

WARNING: No checks on arguments are performed!

**Usage**

```
decision(tau,null,alt=NA,penalty="MBIC",n=0,diffparam=1,pen.value=0)
```

**Arguments**

- `tau` A numeric value or vector specifying the proposed changepoint location(s).
- `null` The value of the null test statistic. If tau is a vector, so is null. If the test statistic is already known (i.e. doesn’t have null and alternative components), replace the null argument with the test statistic.
- `alt` The value of the alternative test statistic (at tau). If tau is a vector, so is alt. If the test statistic is already known, then it is used in replacement of the null argument and the alternative should not be specified (default NA to account for this)
- `penalty` Choice of "None", "SIC", "BIC", "MBIC", "AIC", "Hannan-Quinn", "Asymptotic" and "Manual" penalties. If Manual is specified, the manual penalty is contained in the pen.value parameter. If Asymptotic is specified, the theoretical type I error is contained in the pen.value parameter. The predefined penalties listed DO count the changepoint as a parameter, postfix a 0 e.g."SICO" to NOT count the changepoint as a parameter.
- `n` The length of the original data, required to give sensible "no changepoint" output.
- `diffparam` The difference in the number of parameters in the null and alternative hypotheses, required for the SIC, BIC, AIC, Hanna-Quinn and possibly Manual penalties.
- `pen.value` The theoretical type I error e.g.0.05 when using the Asymptotic penalty. The value of the penalty when using the Manual penalty option - this can be a numeric value or text giving the formula to use. Available variables are, n=length of original data, null=null likelihood, alt=alternative likelihood, tau=proposed changepoint, diffparam=difference in number of alternative and null parameters.

**Details**

This function is used to test whether tau is a true changepoint or not. This test uses the null-alternative as the test statistic and performs the test where the null hypothesis is no change point and the alternative hypothesis is a single changepoint at tau. The test is (null-alt)>=penalty, if TRUE then the changepoint is deemed a true changepoint, if FALSE then n (length of data) is returned.
If the test statistic is already known then it replaces the null value and the alternative is not required (default NA). In this case the test is null>=penalty, if TRUE then the changepoint is deemed a true changepoint, if FALSE then n (length of data) is returned.

This function is exported for developer use only. It does not perform any checks on inputs and is included for convenience and speed for those who are developing their own cost functions.

Value

A list is returned with two elements, cpt and pen.

**cpt**  
If tau is a single value then a single value is returned: Either the value of the true changepoint location or n (length of data) if no changepoint is found.  
If tau is a vector of length m then a vector of length m is returned: Each element is either the value of the true changepoint location or n (length of data) if no changepoint is found. The first element is for the first value of tau and the final element is for the final value of tau.

**pen**  
The numeric value of the penalty used for the test(s).

Author(s)

Rebecca Killick

References


See Also

cpt.mean,cpt.var,cpt.meanvar

Examples

```r
# Example of finding a change
out=c(100,765.1905,435.6529) # tau, null, alt
decision(out[1],out[2],out[3],penalty="SIC",n=200,diffparam=1) # returns 100 as a true changepoint

# Example of no change found
out=c(53,-22.47768,-24.39894) # tau, null, alt
decision(out[1],out[2],out[3],penalty="Manual",n=200,diffparam=1,pen.value="2*log(n)")
```
**ftse100**

*FTSE 100 Daily Returns: 2nd April 1984 – 13th September 2012*

**Description**

This dataset gives the daily returns \((c_{t+1}/c_t - 1)\) of the UK FTSE 100 index from 2nd April 1984 until the 13th September 2012.

**Usage**

ftse100

**Format**

A matrix of dimension 7187 x 2 where the first column is the Date and the second column is the Daily Return.

**Source**

Yahoo! Finance

---

**HC1**

*G+C Content in Human Chromosome 1*

**Description**

This dataset gives the G+C content in 3kb windows along the Human Chromosome from 10Mb to 33Mb (no missing data).

**Usage**

HC1

**Format**

A vector of length 23553.

**Source**

Lai2005fig3

Normalized glioblastoma profile for chromosome 13

Description
This dataset is taken from Lai W, Johnson MJ, Kucherlapati R, Park PJ, Bioinformatics, 2005. The paper states that the original source of the data is from Bredel et al. (2005). The data is Chromosome 13 in GBM31.

Usage
Lai2005fig3

Format
A matrix of dimensions 797 x 5. The columns are Spot, CH, POS.start, POS.end, GBM31.

Source
http://compbio.med.harvard.edu/Supplements/Bioinformatics05b/Profiles/Chrom_13_GBM31.xls

Lai2005fig4

Normalized glioblastoma profile for an excerpt of chromosome 7, the EGFR locus.

Description
This dataset is taken from Lai W, Johnson MJ, Kucherlapati R, Park PJ, Bioinformatics, 2005. The paper states that the original source of the data is from Bredel et al. (2005). The data is an excerpt of chromosome 7 in GBM29 from 40 to 65 Mb.

Usage
Lai2005fig4

Format
A matrix of dimensions 193 x 5. The columns are Spot, CH, POS.start, POS.end, GBM31.

Source
http://compbio.med.harvard.edu/Supplements/Bioinformatics05b/Profiles/Chrom_7_from40_to65Mb_GBM29.xls
Description

Generic function

Usage

ncpts(object)

Arguments

object Depending on the class of object depends on the method used (and if one exists)

Details

Generic Function

Value

Depends on the class of object, see individual methods

Author(s)

Rebecca Killick

See Also

ncpts-methods

Examples

x=new("cpt") # new cpt object
ncpts(x) # returns the number of changepoints (i.e. length of the cpts slot in x minus 1)
nseg

Generic Function - nseg

Description

Generic function

Usage

nseg(object)

Arguments

object

Depending on the class of object depends on the method used (and if one exists)

Details

Generic Function

Value

Depends on the class of object, see individual methods

Author(s)

Rebecca Killick

See Also

nseg-methods

Examples

x=new("cpt") # new cpt object
nseg(x) # returns the number of segments (i.e. length of the cpts slot)
Description

Implements the PELT method for identifying changepoints in a given set of summary statistics for
a specified cost function and penalty.

This function is called by cpt.mean, cpt.var and cpt.meanvar when method="PELT". This is not
intended for use by regular users of the package. It is exported for developers to call directly for
speed increases or to fit alternative cost functions.

WARNING: No checks on arguments are performed!

Usage

PELT(sumstat, pen = 0, cost_func = "norm.mean", shape = 1, minseglen = 1)

Arguments

sumstat  A matrix containing the summary statistics of data within which you wish to
         find a changepoint. Currently assumes 3 columns and uses the number of rows
         as the length of the data +1 (initial value of 0).
pen  Default choice is 0, this should be evaluated elsewhere and a numerical value
      entered. This should be positive - this isn’t checked but results are meaningless
      if it isn’t.
cost_func  The friendly name of the cost function to be called in C. If using your own cost
         function, this must be the name of the C function to use.
shape  Only required for cost_func="Gamma", default is 1. Must be a positive value,
         this isn’t checked.
minseglen  Positive integer giving the minimum segment length (no. of observations be-
         tween changes), default is 1. No checks are performed on the input value so it
         could be larger than feasible to have changes in the data.

Details

This function is used as a wrapper function to implement the PELT algorithm in C. It simply creates
the necessary worker vectors, ensures all inputs are the correct type, and passes everything to the C
function.

This function is exported for developer use only. It does not perform any checks on inputs (other
than type coercion) and is simply a wrapper function for the C code.

Value

A list is returned with elements:

lastchangepoints  Vector of length n containing the last changepoint prior to each timepoint.
penalty_decision

cpts Ordered list of optimal number of changepoints ending with n.
lastchangelike Vector of length n containing the likelihood of the optimal segmentation up to each timepoint.
ncpts Number of changes identified.

Author(s)
Rebecca Killick

References

See Also
cpt.mean, cpt.meanvar, plot-methods, cpt

Examples
#This function should only be used by developers, see its use in cpt.mean, cpt.var and cpt.meanvar.

penalty_decision

Penalty Decision Function - Only intended for developer use.

Description
Evaluates the arguments to give a numeric value for the penalty.
This function is called by cpt.mean, cpt.var and cpt.meanvar. This is not intended for use by regular users of the package. It is exported for developers to call directly for speed increases or to fit alternative cost functions.
WARNING: No checks on arguments are performed!

Usage
penalty_decision(penalty, pen.value, n, diffparam, asymcheck, method)
Arguments

**penalty**  
Choice of "None", "SIC", "BIC", "MBIC", "AIC", "Hannan-Quinn", "Asymptotic" and "Manual" penalties. If Manual is specified, the manual penalty is contained in the pen.value parameter. If Asymptotic is specified, the theoretical type I error is contained in the pen.value parameter. The predefined penalties listed DO count the changepoint as a parameter, postfix a 0 e.g."SIC0" to NOT count the changepoint as a parameter.

**pen.value**  
The theoretical type I error e.g.0.05 when using the Asymptotic penalty. The value of the penalty when using the Manual penalty option - this can be a numeric value or text giving the formula to use. Available variables are, n=length of original data, null=null likelihood, alt=alternative likelihood, tau=proposed changepoint, diffparam=difference in number of alternative and null parameters.

**n**  
The length of the original data, required to give sensible "no changepoint" output.

**diffparam**  
The difference in the number of parameters (degrees of freedom) when a change is added, required for the SIC, BIC, AIC, Hanna-Quinn and possibly Manual penalties. Do NOT include the changepoint when calculating this number as this is automatically added.

**asymcheck**  
A text string which translates to the asymptotic formula for a specific cost function. Currently implemented values are: mean.norm, var.norm, meanvar.norm, reg.norm, var.css, mean.cusum, meanvar.gamma, meanvar.exp, meanvar.poisson.

**method**  
Method used as a text string, see *cpt.mean* for further details.

Details

This function takes the text string input and converts it to a numerical value for the specific length of data specified by n.

This function is exported for developer use only. It does not perform any checks on inputs and is included for convenience and speed for those who are developing their own cost functions.

Value

The numeric value of the penalty.

Author(s)

Rebecca Killick

References


See Also

cpt.mean, cpt.var, cpt.meanvar

Examples

```r
# Example of finding a change
out <- c(100, 765.1905, 435.6529) # tau, null, alt
decision(out[1], out[2], out[3], penalty="SIC", n=200, diffparam=1) # returns 100 as a true changepoint

# Example of no change found
out <- c(53, -22.47768, -24.39894) # tau, null, alt
decision(out[1], out[2], out[3], penalty="Manual", n=200, diffparam=1, pen.value="2*log(n)")
```

---

**Description**

Generic function

**Usage**

`seg.len(object)`

**Arguments**

- `object` Depending on the class of `object` depends on the method used (and if one exists)

**Details**

Generic Function

**Value**

Depends on the class of `object`, see individual methods

**Author(s)**

Rebecca Killick

**See Also**

`seg.len-methods`
Examples

x=new("cpt") # new cpt object
seg.len(x) # returns the length of each segment in the data (i.e. no. of obs between changepoints)

Description

This dataset gives the significant wave heights from buoy c44137 obtained from the Fisheries and Oceans Canada, East Scotian Slop. The data are taken at hourly intervals from January 2005 until September 2012.

Usage

wave.c44137

Format

A vector of length 63651.

Source

Index

* changepoint
  changepoint-package, 2
* cpt
  ncpts, 21
  nseg, 22
  seg.len, 26
* datasets
  ftse100, 19
  HC1, 19
  Lai2005fig3, 20
  Lai2005fig4, 20
  wave.c44137, 27
* methods
  BINSEG, 3
  class_input, 5
  cpt.mean, 6
  cpt.meanvar, 9
  cpt.var, 13
  decision, 16
  ncpts, 21
  nseg, 22
  PELT, 23
  penalty_decision, 24
  seg.len, 26
* models
  BINSEG, 3
  class_input, 5
  cpt.mean, 6
  cpt.meanvar, 9
  cpt.var, 13
  decision, 16
  ncpts, 21
  nseg, 22
  PELT, 23
  penalty_decision, 24
* segmentation
  changepoint-package, 2
* ts
  BINSEG, 3
  class_input, 5
  cpt.mean, 6
  cpt.meanvar, 9
  cpt.var, 13
  decision, 16
  PELT, 23
  penalty_decision, 24
  seg.len, 26
  wave.c44137, 27