changepoint.online: An R Package for Online Changepoint Analysis

Andrew Connell, Rebecca Killick, David Matteson

2018-08-14

Abstract

One of new the key challenges in changepoint analysis is the need for an online approach. The **changepoint.online** package has been developed to provide users with a choice of changepoint search methods to use in conjunction with a given changepoint method and in particular provides an implementation of the recently proposed PELT algorithm much like the **changepoint** package does. The **changepoint.online** in fact has the same functionality and can produce the same results for offline data as the **changepoint** package however, it can also analysise online data. This article describes the methodology of the new online approachs and the key differences between the two methods with simulated and practical examples. Another key change is the *ECP* test statistic is now available, which stems from the **ecp** package.

Keywords: R; exact algorithm; linear cost; online changepoint detection; PELT; ECP; Energy Time Series.

1. Introduction

In recent years, there has been an increase in applications in which data is collected in an online manner, also known as streaming data. In many cases, it is not feasible or advisable to store this data prior to analysis. Thus, there is a growing need to develop analysis methods specifically for the online setting, see for example Gama and Rodrigues (2007). The ability to detect changes in the distributional properties of a time series accurately and efficiently in an online fashion has important implications in many industrial applications, for example in forecasting Koop and Potter (2007), consumer profiling (Whittaker et al., 2007) and fault detection in industrial process control Lai (1995); it is also a critical consideration in (online or offline) processing of time series segments and data storage applications Golab and Özsu (2003).

The problem of online changepoint analysis of time series has been considered extensively in the literature due to its importance in many scientific fields (see Page, 1954; Hawkins et. al, 2003). The changepoint problem for online data streams is typically formulated in a sequential hypothesis test framework in which, as new data arrives, the hypothesis of a change is tested, conditional on the data observed in the past. If a changepoint is found to be statistically significant, the changepoint detection method is "reset", i.e. the changepoint analysis restarts from the next observed data point. As far as we are aware, this methodology is only available in the **cpm** R package Ross (2015) which provides a selection of distributional and nonparametric data assumptions.

A criticism of the sequential resetting or "conditional" formulation of the multiple changepoint problem above is that there is no accumulation of useful information as the data stream progresses: as soon as a change is found, the analysis is restarted as if the arriving data is an independent dataset, and thus previous information is not retained for assessing future changepoint locations. In particular, any errors made by the changepoint detection method early in the data stream propagate such that they affect any inference made about subsequent changes in the stream, adding uncertainty to those potential change locations. In addition, since any new data after a change are considered independent, detected changes in the past cannot be re-evaluated to improve accuracy in location estimation. This is not conducive to a streaming data environment where both speed and accuracy are valued.

Recently, Killick, Fearnhead, and Eckley (2012) proposed a changepoint search algorithm, named the Pruned Exact Linear Time (PELT) algorithm, which is shown to be exact and whose complexity is linear in the number of data points. Since the PELT method achieves the benefits of both exactness and computational efficiency, it has thus been shown to perform particularly well for applications in which the analysis of large

datasets is required. A further development for nonparametric cost functions which provides an approximate pruning step, e-cp3o, is given in James and Matteson (2015). These algorithms are available in the popular changepoint and ecp packages on CRAN.

Our motivation for considering these algorithms is that, since they reassess the locations of potential changepoints each time new data arrives, the propagation of errors in detected changes is circumvented when performing inference about changes later in the stream. Furthermore, the exact (for PELT) and minimal computational complexity of the algorithms ensures these techniques find changes in data streams in an efficient and accurate fashion. Due to the dynamic programming in these algorithms they are inherently online in nature but were only available in R packages **changepoint** Killick, Haynes, and Eckley (2016) and **ecp** James and Matteson (2014) which require the full data for analysis. This package mirrors the functionality of the changepoint and ecp packages in terms of functionality but for provided initialisation functions and update functions required in an online setting. This paper describes the **changepoint.online** package, available for R from the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/package=changepoint.online.

2. Online Changepoint Detection

This paper will not specifically detail the single and multiple changepoint methods discussed by Killick and Eckley (2014), but instead focus on the new online method that has been developed. Furthermore, the initialisation function will be given less emphasis than the update function as the new online method is more clearly explained by the update as this is the primary reason for the packages development.

More formally, let us assume we have an initial ordered sequence of data, $y_{1:n} = (y_1, ..., y_n)$ and an update ordered sequence $y_{n+1:m} = (y_{n+1}, ..., y_m)$ where we note $n, m \in \mathbb{N}$ and $m > n$. A changepoint is said to occur within the initial set when there exists a time, $\tau_\alpha \in 1, ..., n-1$, such that the statistical properties of $y_1, ..., y_{\tau_\alpha}$ and $y_{\tau_\alpha+1}, ..., y_n$ are different in some way. Clearly this can be extended to a changepoint within the updated set and we will denote this τ_β. We will denote the set of all changepoints as $\tau_{1:p} = (\tau_1, ..., \tau_p)$ where the number of changepoints must be between 1 and $n-1$ inclusive for initialisation and n and $m-1$ inclusive for update and thus between 1 and $m-1$ inclusive when we consider the total length. Note that this means that p, the total number of changepoints, can and likely will increase as more updates are added. We define $\tau_0 = 0$ and $\tau_{p+1} = n$ or m depending on whether we are discussing an initialisation or update respectively , and assume that the changepoints are ordered so that $\tau_i < \tau_j$ if, and only if, $i < j$. Consequently the p changepoints will split the data into $p+1$ segments, with the i^{th} segment containing data $y_{(\tau_{i-1}+1)} : \tau_i$. Each segment will be summarized by a set of statistical parameters, whether that be a change in mean, variance or something else. The parameters associated with the i^{th} segment will be denoted θ_i, ϕ_i, where ϕ_i is a (possibly null) set of nuisance parameters and θ_i is the set of parameters that we believe may contain changes. Typically we want to test how many segments are needed to represent the data, i.e., how many changepoints are present and estimate the values of the parameters associated with each segment.

In the following subsections, we will focus on multiple changepoint detection for brevity. The single changepoint detection case can be thought of as a special case of the multiple and thus the multiple changepoint detection method generalises all cases.

2.1. PELT.online

The PELT algorithm already calculates quantities in an online fashion. Therefore, the implimentation only need concern itself with fixing a $\mathcal{O}(1)$ computational time per calculation and allocating a set amount of memory.

Both the initialisation and update functions pass the information in to PELT.c, this is the new online PELT algorithm. For the initialisation, it starts with empty vectors while for the update some information is carried over. Specifically, for any update we must carry over the last summary statistic row as each point, with

the exception of the first initialisation point, looks back one piece of data. Now when considering viable changepoint locations, there are different cases to note. Firstly, there isdetecting a first changepoint. This paper will not discuss this trivial case, as the methodology is the same as the offline approach and is in fact the same as the resetting case. The other cases are when a new changepoint is found and the effect this has on the previous changepoint whether that be the previous changepoint disappearing, moving or remaining the same. Furthermore, due to the nature of the online method in the case of the previous changepoint disappearing, it is thought of as resetting as the new changepoint becomes τ_1. This leaves the two cases of the previous changepoint staying in its current position or moving to a different position. It should be noted that if the previous changepoint moves it must be to a point between the minimum segment length and the new changepoint that has been detected.

The mathematics behind the PELT.online algorithm is in fact very similar to the offline method with some small but crucial changes. Firstly, much like for the PELT algorithm each segment has its specific cost and we use the previously proven proposition for offline PELT of $F(t) + C(y_{(t+1):s}) > F(s)$ where t is the previous changepoint, s is the next changepoint and $C()$ is the specific cost function. Using this proposition, then it can be shown that for a future time T, the last changepoint cannot be at any location $t < s$. This means that these locations can be removed from the minimisation search and this is the basis of the online approach. Since this pruning step excludes locations which cannot occur as changepoints, the changepoint search remains exact but reduces in computational cost from $\mathcal{O}(n^2)$ to be linear in the number of data. This is related to number of changepoints detected so computational complexity does increase linearly as more changepoints are detected. At the end of this updating of the changepoints, for each iteration of the PELT.online function, the storage values are all updated, for instance ndone (the previous points analysed) and nupdate (the points to be analysed in the next update).

2.2. ECP.online

The "ECP" test.stat leads to a seperate C++ file and this function is an online variant of the test statistic used in the **ecp** package. The method is similar to PELT due to both methods looking back through vectors and using a cost function and penalty to make decisions. Unlike the PELT.online method, which must pull forward the final row of the summary statistic, the ECP approach must recover the last delta rows from the summary statistic. Although the original preset for delta is 29 in the original **ecp** pacakge, values as low as 2 will now still yield as accurate results. Therefore, although being a slightly more computationally inefficient method than PELT, the difference for most users will be negligiible.

The fundamental difference in the new ECP method is that the distance vectors are kept in the delta overlap, by keeping these vectors we are able to first of all recover and calculate whether any changepoints will occur in the last segment length of the initialised data and the first segment length of the updated data. All further updated calculations in the new length beyond the delta overlap, are treated in an iterative process much like the original ECP method. The distance values around the individual changepoints are stored in a similar way to the PELT algoritm. Therefore much like the PELT algorithm the computational cost is linear with respect to the number of changepoints found. Furthermore, nupdate is replaced with newlength and ndone is replaced with oldlength. This is only done so to minimise confusion for users. The only other recovered information is the last row of the cumulative sum so as the update cumulative sum can continue to be calculated. Once all the initialised or updated vectors have been input or calculated, the possible changepoint locations are then scored using a Goodness of Fit model much like the cost functions for PELT. These individual scores are then compared against a threshold as well as each other and the algorithm decides the most appropriate changepoint locations. Unlike PELT, the user can force the ECP algorithm to have a maximum number of changepoints it will be allowed to pick. Caution is advised when using this feature.

3. Introduction to the package and the 'ocpt' class

The **changepoint** package introduced a new object class called 'cpt' to store changepoint analysis objects. This section provides an introduction to the new 'ocpt' structure in **changepoint.online** which follows on

from the 'cpt' class previously introduced in Killick and Eckley (2014). The new 'ocpt' class contains the functions within the 'cpt' class however, it also combines them with new functions that are approrpiate to the online method.

Each of the core functions outputs an object of the 'ocpt' S4 class. The class has been constructed such that the 'ocpt' object contains the main features required for the online changepoint analysis and future summaries. Several objects, although masked from the user and the summary, are filled within slots as they are required for the update functions. Each of the objects are stored within a slot entry in the 'ocpt' class. The slots within the class are:

- *sumstat* – a matrix containing the summary statistics for the data;
- *cpttype* – characters describing the type of changepoint sought e.g., mean, variance;
- *method* – characters denoting the single or multiple changepoint search method applied.
- *test.stat* – characters denoting the test statistic, i.e., assumed distribution/distribution - free method;
- *pen.type* – characters denoting the penalty type;
- *pen.value* – the numeric value of the penalty used in the analysis. Unlike in previously, pen.value also can be calculated using Average Run Length;
- *cpts* – vector giving the estimated changepoint locations ending in the length of the time series in the data.set slot;
- *ncpts.max* – the numeric maximum number of changepoints searched for, e.g., 1, 5, Inf;
- *param.est* – a list of parameters where each element in the list is a vector of the estimated numeric parameter values for each segment;
- *date* – the system time / date when the analysis was performed.
- *version* - Version number of the package used when the analysis was run.
- *lastchangelike* - vector containing the likelihood of the optimal segmentation up to each timepoint.
- *lastchangecpts* - vector containing the last changepoint prior to each timepoint.
- *numchangecpts* - stores the current number of changepoints detected.
- *checklist* - vector of locations of the potential last changepoint for next iteration. Used only for update.
- *ndone* - length of the time series when analysis begins.
- *nupdate* - length of the time series to be analysed in this update.
- *cost_func* - the cost function used to decide possible changepoints.
- *shape* - only used when cost_func is the gamma likelihood. Otherwise 1.

Currently the method slot is masked from the user as PELT is the only method implemented. However, more methods are being added and at that point this option will be unmasked. Slots like version, although have nothing to do with the algorithm, are useful for continuity for the user who is running updates over a significant period of time.

The **changepoint.online** package is designed so that users who are already comfortable with the **changepoint** Killick, Haynes, and Eckley (2016) package will easily be able to use the online version. Therefore, much like with the **changepoint** package, we have created accessor and replacement functions to control the access and replacement of slots rather than using the @ accessor. Although the @ symbol is still available to use, from the feedback from the **changepoint** package users, it was shown that the accessor and replacement functions aided ease-of-use for those unfamiliar with S4 classes. The accessor functions are simply the slot names. For example data.set(x) displays the vector of data contained within the 'cpt' object x. The class slots are automatically populated with the correct information obtained from the completed analysis. Further

demonstration of how the accessor and replacement functions work in practice are given in the examples within each section and can be found with the *man* section of the package itself.

All the functions described above are related to the 'ocpt' class within the **changepoint.online** package. However a further class 'ecp.ocpt' exists which contains all of the above as well as some additional information which is required in the case of the '*ECP*' test statistic. This information includes:

- *GofM* - goodness of fit model.
- *delta* - the window size used to calculate the calculate the complete portion of our approximate test statistic.
- *alpha* - the moment index used for determining the distance between and within segments.
- *verbose* - a flag indicating if status updates should be printed.

This class is created seperately rather than masking these options in cases when they are not needed to ensure that each process is at its optimal efficiency.

3.1 Methods within the 'ocpt' class

The methods associated with the 'ocpt' class are the same as the 'cpt' class discussed by Killick and Eckley (2014). These are summary, show, param, plot and logLik. Although these may not seem the exact same, print has been renamed to show but has the same functionality. The summary and show methods display standard information about the 'ocpt' object. The summary function displays a synopsis of the results from the analysis including number of changepoints and, where this number is small, the location of those changepoints. The summary function is called by all the initialisation and update functions within the package. In contrast, the show function prints details pertaining to the S4 class including slot names and when the S4 object was created.

Having performed a changepoint analysis, it is often helpful to be able to plot the changepoints on the original data to visually inspect whether the estimated changepoints are reasonable. To this end we include a plot method for the 'ocpt' class. Furthermore, there is a new *ocpt.plot* function which produces a animated plot so as the data is plotted the last changes are calculated and clearly marked. This new function specifically uses this method. The method adapts to the assumed type of changepoint, providing a different output dependent on the type of change. For example, a change in variance is denoted by a vertical line at the changepoint location whereas a change in mean is indicated by horizontal lines depicting the mean value in different segments. These rules outlined are the same irrespective of the graph being animated or a single plot.

Similarly once a changepoint analysis has been conducted one may wish to retrieve the parameter values for each segment or the log likelihood for the fitted data. These can be obtained using the standard param and logLik generics; examples are given in the code detailed below.

The following sections explore the use of the core functions within the **changepoint.online** package. We begin in Section 4 by demonstrating the key steps to a changepoint analysis via the ocpt.mean functions and we will discuss both the initialisation and update function. Sections 5 and 6 utilize the steps in the change in mean analysis to explore changes in variance and both mean and variance respectively. Section 7 will go on to discuss the e.cp3o functions and there applications.

4. Changes in Mean: The ocpt.mean functions

Within the **changepoint.online** package all change in only mean methods are accessed using the *ocpt.mean* functions. The functions are structured as follows:

ocpt.mean.initialise(data,penalty="Manual",pen.value=length(data),Q=5,test.stat="Normal",class=TRUE, param.estimates=TRUE,shape=1,minseglen=1,alpha=1,verbose=FALSE)

It is worth noting that American English versions of all the functions exist that are written as *ocpt.mean.initialize()* exist but are computationally identical to the British English version. The arguments for both functions are:

- *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" 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. Note the general preset for a Manual penalty is the average run length of the data.
- *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 alternatve and null parameters.
- *Q* - The maximum number of changepoints to search for using the "ECP" test.stat through the recursive loop. This makes no changes for any of the other test statistics.
- *test.stat* - The assumed test statistic / distribution of the data. Currently only "ECP" "Normal", "Exponential", "Gamma" or "Poisson" supported.
- *class* - Logical. If TRUE then an object of class *ocpt* 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* - Only used when cost function is the gamma likelihood.
- *minseglen* - Positive integer giving the minimum segment length (no. of observations between changes), default is the minimum allowed by theory. Note that when the "ECP" test.stat is used then the input value for minseglen is one less than the delta value used.
- *alpha* - The moment index used for determining the distance between and within segments. only used in "ECP" case.
- *verbose* - Allows the user to see certain aspects of the internal processes. Only for "ECP" case.

The function structure for ocpt.mean.updated is:

ocpt.mean.update(previousanswer, newdata)

with arguments:

- *previousanswer* - A S4 class. This will be the output from the *ocpt.mean.initialise()* function.
- *newdata* - A vector, ts object or matrix containing the new data within which you wish to find a changepoint. If data is a matrix, each row is considered a separate dataset.

The update function has far fewer options as a key assumption when updating is that certain decisions such as the penalty cannot be changed as it would invalidate earlier output results. The user choices are thus carried forward and the new data can then be treated as an extension of the previous result.

The remainder of this section gives a worked example exploring how to identify a change in mean.

4.1. Example: Changes in mean

We now describe the general structure of an online changepoint analysis using the **changepoint.online** package. We begin by demonstrating the various possible stages within a change in mean analysis. To this end

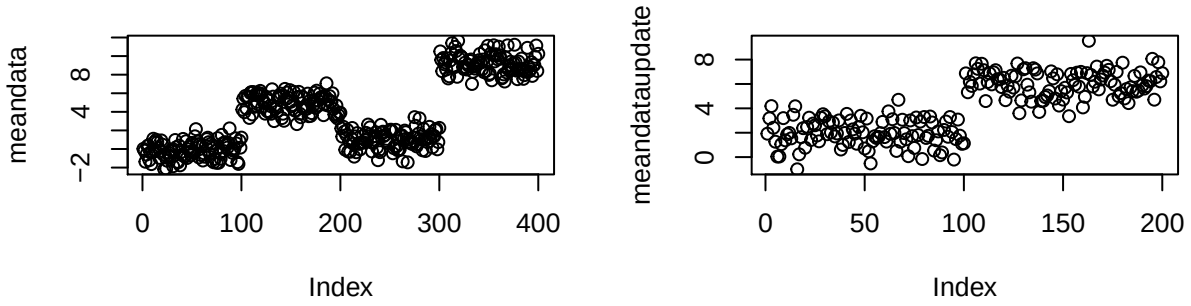


Figure 1: 1(A) & 1(B)

we simulate two datasets (*meandata*) and (*meandataupdate*) of length 400 and 200 respectively. *meandata* has changepoints at 100, 200 and 300 while *meandataupdate* has only one change at 100, however this should produce two results when added as an update at 400 and 500. The sequence will have six segments and the means for each segment are 0, 5, 1, 9, 2, 6.

```
library("changepoint.online")
set.seed(10)
meandata <- c(rnorm(100, 0, 1), rnorm(100, 5, 1), rnorm(100, 1, 1),rnorm(100, 9, 1))
meandataupdate <- c(rnorm(100, 2, 1), rnorm(100, 6, 1))
```

Imagine that we have been presented with the *meandata* dataset and are asked to perform a changepoint analysis and have been told new data (the *meandataupdata*) will be coming live at a later point once the initial analysis has been done. The first question we aim to answer is "Is there a change within the data and if so how many changes are there?". Our first choice in answering this question is whether we wish to consider a single change or whether multiple changes are plausible. From a visual inspection of the data in Figure 2, we suspect multiple changes in mean may exist within the two lots of data. The challenge in multiple changepoint detection is identifying the optimal number and location of changepoints as the number of solutions increases rapidly with the size of the data and also ensuring with any additional updates that the optimal changepoints are correctly chosen. In this initial example where $n = 400$, we have 399 possible solutions for a single changepoint, for two changes there are 79401 possible solutions and this is not taking into account that we do not know how many changes there are! So we must think of our possible changepoint locations as $\binom{n}{Q}$ where n is the total number of datapoints and Q is the maximum number of changepoints the user wishes the algorithm to search for. As such it is clearly desirable to use an efficient method for searching the large solution space. Either of the two search methods could be used to detect these changes. For this

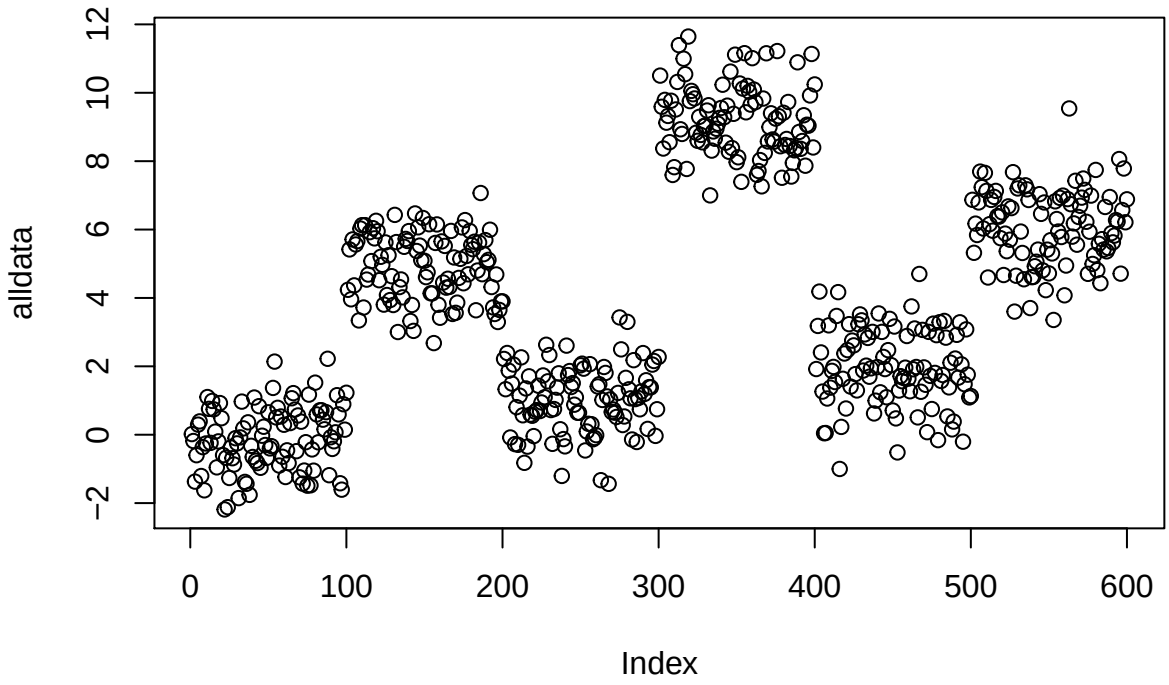
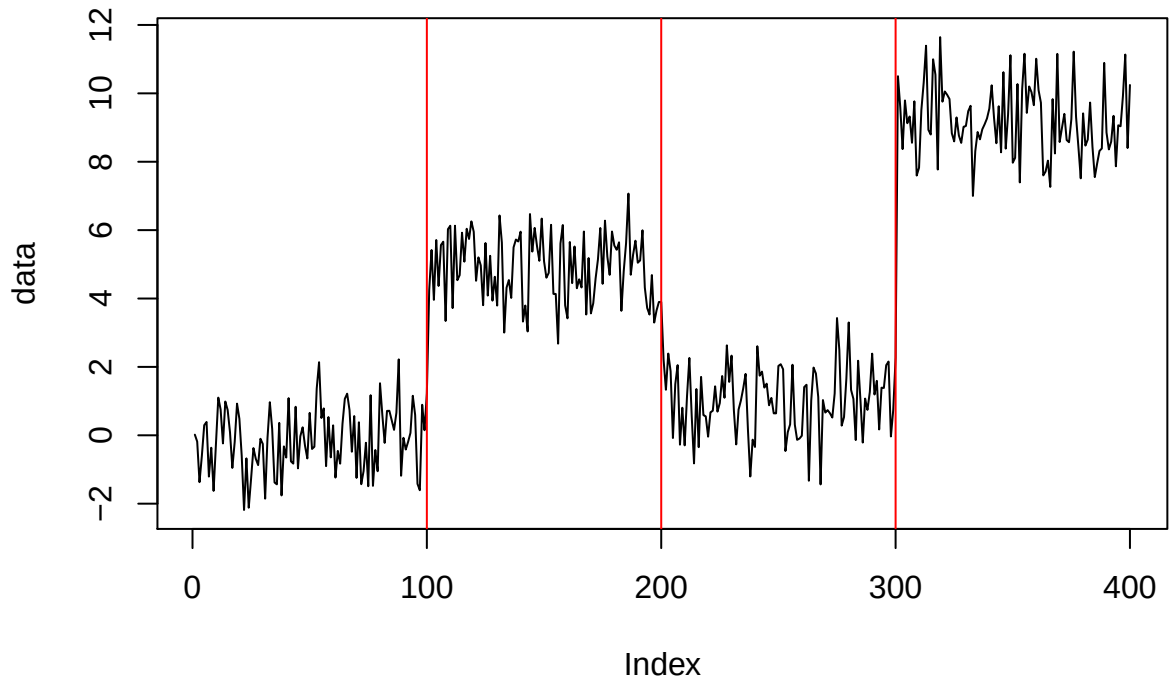


Figure 2: All data

example we will compare the PELT and ECP. For now we will assume that the dataset is independent and Normally distributed and consider an alternative towards the end of this section.

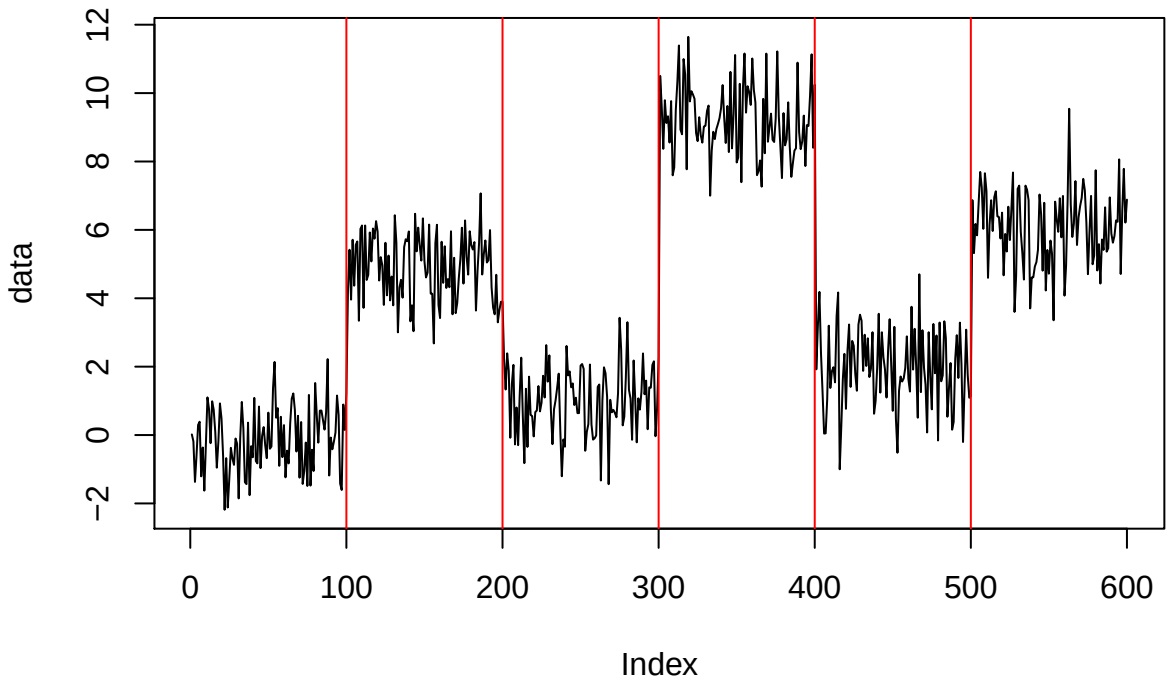
```
meanpeltinitialise <- ocpt.mean.initialise(meandata)
meanpeltinitialise
#> Class 'ocpt' : Changepoint Object
#>         ~~   : S4 class containing 20 slots with names
#>                cpttype date version sumstat method test.stat pen.type pen.value minseglen cpts ncpts.
#>
#> Created on  : Tue Aug 14 00:20:32 2018
#>
#> summary(.)  :
#> ----------
#> Created Using changepoint.online version 0.1
#> Changepoint type      : Change in mean
#> Method of analysis    : PELT
#> Test Statistic  : Normal
#> Type of penalty       : Manual with value, 400
#> Minimum Segment Length : 1
#> Maximum no. of cpts   : Inf
#> Changepoint Locations : 100 200 300
#> ndone              : 2
#> nupdate            : 398
plot(meanpeltinitialise,data=meandata, type = "l", xlab = "Index",cpt.width = 4)
abline(v=cpts(meanpeltinitialise),col="red")
```



```
cpts(meanpeltinitialise)
#> [1] 100 200 300
```

The updated function will now be added to this plot and the cpts will be shown.

```
meanpeltupdate <- ocpt.mean.update(meanpeltinitialise,meandataupdate)
plot(meanpeltupdate,data=alldata, type = "l", xlab = "Index",cpt.width = 4)
abline(v=cpts(meanpeltupdate),col="red")
```




```
cpts(meanpeltupdate)
#> [1] 100 200 300 400 500
```

As users can check this shows the compuational period is the same for any intialisation and update ths showing it is an online method. Now we must compare this to the ecp test statistic choice. The only difference between using *cpt.mean* on the whole data and using the new online approach is that the new online method is more computationally efficient. The major benefit of the ECP process is the added functionality of multivariate data. The process of updating can be thought of in relatively the same way however the returned summary is slighty different

```
matrixdata <- matrix(c(rnorm(100,5,10),rnorm(100,25,1),rnorm(100,5,10),rnorm(100,25,1)),ncol=2)
ecpmeaninit <- ocpt.mean.initialise(matrixdata, test.stat = "ECP")
ecpmeaninit
#> Class 'ocpt' : Changepoint Object
#> summary(.)  :
#> ----------
#> Number of Changepoints   : 1
#> Estimate Locations       : 98
```

```
#> Goodness of Fit Model    : 14.61535 19.45456 25.42187 32.73556 38.6008
#> Delta                    : 2
#> Alpha                    : 1
#> Verbose                  : FALSE
#> Number of Data points    : 200
#> Calculation Time         : 0.022
```

Thus far we have only considered a simulated example. In the next section we apply the *ocpt.mean* functions to some Glioblastoma data previously analysed by Lai et al. (2005) .

4.2. Case Study: Glioblastoma

Lai et al. (2005) compare different methods for segmenting array comparative genomic hybridization (aCGH) data from Glioblastoma multiforme (GBM), a type of brain tumor. These arrays were developed to identify DNA copy number alteration corresponding to chromosomal aberrations. High-throughput aCGH data are intensity ratios of diseased vs. control samples indexed by the location on the genome. Values greater than 1 indicate diseased samples have additional chromosomes and values less than 1 indicate fewer chromosomes. Detection of these aberrations can aid future screening and treatments of diseases. The data is replicated in the **changepoint.online** package for ease. Following Lai et al. (2005) we fit a Normal distribution with a piecewise constant mean using a likelihood criterion. Figure 2 demonstrates that PELT.online (with default penalty) gives the same segmentation as the CGHseg method from Lai et al. (2005).

```
data("Lai2005fig4", package = "changepoint.online")
Lai.default.initialisation <- ocpt.mean.initialise(Lai2005fig4[1:100, 5],pen.value = 10)
Lai.default.update <- ocpt.mean.update(Lai.default.initialisation,Lai2005fig4[101:191, 5])
cpts(Lai.default.update)
#> [1]  81  85  89  96 123 133
coef(Lai.default.update)
#> $mean
#> [1]  11.44168  26.76746  39.53044  54.96388  75.05496  96.42812 126.81278
```

5. Changes in Variance: The ocpt.var functions

6. Changes in Mean and Variance: The ocpt.meanvar functions

7. Summary

The unique contribution of the **changepoint.online** package is that the user has the ability to update previously existed data in a truly online method while still looking back through the whole data. The package currently contains two unique styles of doing this: ECP and PELT and this paper has described and demonstrated some differences between these approaches. Furthermore, this paper has given some examples and uses of the different initialisation and update functions whether that be for changes in mean and/or variance using distributional or distribution-free assumptions. The package has brought both the **ecp** and **changepoint** packages into an online perspective while not losing accuracy or efficiency. As such the **changepoint.online** package is useful in the modern approach of data analysis.

Acknowledgements

The authors wish to thank Ben Norwood for helpful insight and discussions during the project. A. Connell acknowledge financial support from Google via the Google Summer of Code stipend.

References

Gama, João, and Pedro Pereira Rodrigues. 2007. "Stream-Based Electricity Load Forecast." Edited by Joost N. Kok, Jacek Koronacki, Ramon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron. Berlin, Heidelberg: Springer Berlin Heidelberg, 446–53.

Golab, Lukasz, and M. Tamer Özsu. 2003. "Issues in Data Stream Management." *SIGMOD Rec.* 32 (2). New York, NY, USA: ACM: 5–14. https://doi.org/10.1145/776985.776986.

James, Nicholas A., and David S. Matteson. 2014. "ecp: An R Package for Nonparametric Multiple Change Point Analysis of Multivariate Data." *Journal of Statistical Software* 62 (7): 1–25. http://www.jstatsoft.org/v62/i07/.

———. 2015. "Change Points via Probabilistically Pruned Objectives."

Killick, Rebecca, and Idris A. Eckley. 2014. "changepoint: An R Package for Changepoint Analysis." *Journal of Statistical Software* 58 (3): 1–19. http://www.jstatsoft.org/v58/i03/.

Killick, Rebecca, Kaylea Haynes, and Idris A. Eckley. 2016. *changepoint: An R Package for Changepoint Analysis*. https://CRAN.R-project.org/package=changepoint.

Killick, R., P. Fearnhead, and I. A. Eckley. 2012. "Optimal Detection of Changepoints with a Linear Computational Cost." *Journal of the American Statistical Association* 107 (500). Taylor & Francis Group: 1590–8.

Koop, Gary, and Simon M. Potter. 2007. "Estimation and Forecasting in Models with Multiple Breaks." *The Review of Economic Studies* 74 (3). [Oxford University Press, Review of Economic Studies, Ltd.]: 763–89. http://www.jstor.org/stable/4626160.

Lai, Tze Leung. 1995. "Sequential Changepoint Detection in Quality Control and Dynamical Systems." *Journal of the Royal Statistical Society. Series B (Methodological)* 57 (4). [Royal Statistical Society, Wiley]: 613–58. http://www.jstor.org/stable/2345934.

Lai, Weil R., Mark D. Johnson, Raju Kucherlapati, and Peter J. Park. 2005. "Comparative Analysis of Algorithms for Identifying Amplifications and Deletions in Array Cgh Data." *Bioinformatics* 21 (19). Oxford University Press: 3763–70.

Ross, Gordon J. 2015. "Parametric and Nonparametric Sequential Change Detection in R: The cpm Package." *Journal of Statistical Software* 66 (3): 1–20. http://www.jstatsoft.org/v66/i03/.