Parametric and Nonparametric Sequential Change Detection in R: The cpm package

Gordon J. Ross
University of Bristol
gordon.ross03@ic.ac.uk

Abstract

The change point model (CPM) framework introduced in Hawkins, Qiu, and Kang (2003) and Hawkins and Zamba (2005) provides an effective and computationally efficient method for sequentially detecting multiple changes points in streams of Gaussian random variables, when no information is available regarding the parameters of the distribution in the various segments. It has since been extended in various ways by Hawkins and Deng (2010), Ross, Adams, Tasoulis, and Hand (2011) and Ross and Adams (2011b) to allow for fully nonparametric change detection in non-Gaussian sequences, when no knowledge is available regarding even the distributional form of the sequence. Another extension comes from Ross and Adams (2011a) which allows change detection in streams of Bernoulli random variables, again when the values of the parameters are unknown.

This paper describes the R package **cpm**, which provides a fast implementation of all the above change point models. Its usage is illustrated through numerous examples.

Keywords: change detection, sequential analysis, R.

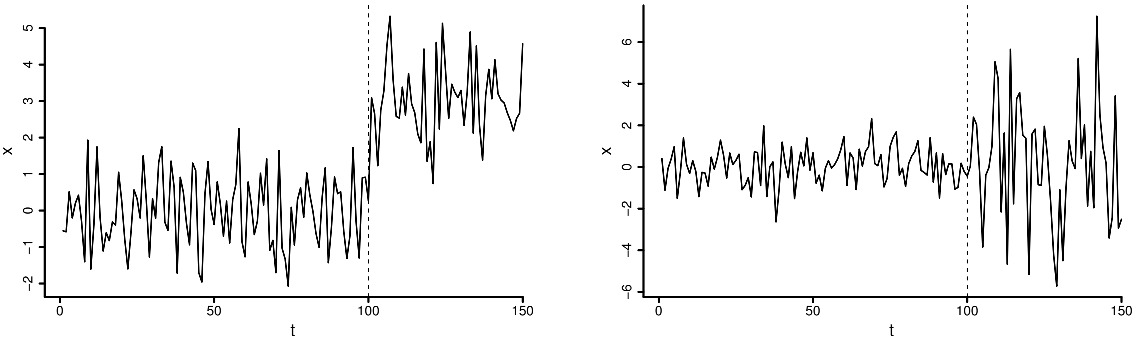
1. Introduction

Many statistical problems require change points to be identified in sequences of data. The usual setting for this is where a sequence of observations $x_1, x_2, \ldots$ is drawn from the random variables $X_1, X_2, \ldots$ and undergo one or more abrupt changes in distribution at the unknown change points $\tau_1, \tau_2, \ldots$. It is usually assumed that the observations are independent between every pair of change points, i.e. the distribution of the sequence can be written as:

$$X_i \sim \begin{cases} F_0 & \text{if } i \leq \tau_1 \\ F_1 & \text{if } \tau_1 < i \leq \tau_2 \\ F_2 & \text{if } \tau_3 < i \leq \tau_4, \\ \ldots \end{cases} \tag{1}$$

where the F_is describe the distribution in each segment. Some simple examples are given in Figures 1a and 1b, which show two sequences of Gaussian random variables which undergo changes in their mean and variance respectively.

Although requiring independence between change-points seems a restrictive assumption, this is not the case since dependence can typically be handled by first modelling any underlying dynamics or drift process, and then performing change detection on the model residuals or



(a) Mean Change

(b) Variance Change

Figure 1: Basic examples of changes to a univariate stream, with the time of the change point superimposed.

one-step ahead prediction errors, both of which should give i.i.d sequences, provided that the model has been correctly fit. For more information on this topic, see Gustafsson (2000). In the remainder of this paper, it is assumed that such modelling has been carried out, and that we are left with sequences of observations which are i.i.d conditional on the change points.

Change detection problems differ based on what is assumed to be known about the F_i distributions. In most practical situations the parameters of these distributions will be unknown, and in many cases there may not even be information available about their distributional form. Recently the change point model (CPM) framework has been developed for change detection in situations where the available information about the F_i distributions is very limited. Several different CPMs have been developed, which incorporate different test statistics to enable changes to be detected in a wide variety of sequences, using both parametric and nonparametric techniques (Hawkins *et al.* 2003; Hawkins and Zamba 2005; Hawkins and Deng 2010; Zhou, Zou, Zhang, and Wang 2009; Ross *et al.* 2011; Ross and Adams 2011b,a).

The purpose of this paper is to describe the **cpm** R package which implements most of these published CPM models for detecting changes in the distribution of discrete-time sequences of univariate random variables. Unlike existing R packages related to change detection, such as **bcp** (Erdman and Emerson 2007) and **changepoint** (Killick and Eckley 2011), the focus of **cpm** is Phase II analysis when the parameters and perhaps distributional form of the observations are unknown. We hope that providing an easy-to-use implementation of the variety of published change point models will be useful to practitioners.

Section 2 of this paper details how to obtain the **cpm** package. Section 3 gives a more detailed overview of the sequential change detection problem. Section 4 provides a general overview of the CPM framework. Finally, the **cpm** package is introduced in Section 5 and an overview of its capabilities is given, along with many examples of use.

2. Package installation

This package can be obtained in two ways. The first is to obtain it from the Comprehensive

R Archive Network (CRAN), at http://cran.r-project.org/web/packages/cpm/index.html. To install the package from CRAN simply start an R session at a computer which has an internet connection, and type: `install.packages('cpm')`.

The package should then download and install automatically. Alternatively, the package can be found on the author's website at http://gordonjross.co.uk. After downloading the `cpm_<version>.tar.gz` file, the package can be installed by loading R and typing:

```
install.packages("cpm_<version>.tar.gz", repos=NULL, source=TRUE)
```

where the first argument should be replaced with the package download location, and `<version>` should be replaced with the version number of the package. Regardless of how the package was obtained, it can be loaded after installation by typing:

```
library("cpm")
```

3. Background information

The change detection problem described by Equation 1 in the previous section has been a lively area of research since the 1950s. Because of the very general nature of the problem, the literature is diverse and spans many fields. In particular, many popular methods have their origin in the quality control community, where the goal is to monitor the output of industrial manufacturing processes and detect faults as quickly as possible (Lai 1995) However there are many other situations where change detection techniques are important, such as identifying copy number variation in genome data (Efron and Zhang 2011), detecting intrusions in computer networks (Tartakovsky, Rozovskii, Blazek, and Kim; 2006), and fitting the multiple regime models which are popular in economics and finance (Bai and Perron 1998).

There are two main types of change detection problem, batch and sequential. In the quality control literature, these are respectively known as Phase I and Phase II detection:

1. **Batch detection** (Phase I): In this case, there is a fixed length sequence consisting of n observations from the random variables $X_1, \ldots, X_n$, and it is required to test whether this sequence contains any change points. This type of change detection is retrospective, meaning that the decision whether a change has occurred at a particular point in the sequence is made using all the available observations, including those which occur later in the sequence. These batch methods work well when there are only a small number of change points, but can be computationally infeasible when larger numbers of change points are present, unless heuristics are used (Inclan and Tiao 1994; Hawkins 2001).

2. **Sequential detection** (Phase II): In this case, the sequence does not have a fixed length. Instead, observations are received and processed sequentially over time. When each observation has been received, a decision is made about whether a change has occurred based only on the observations which have been received so far. If no change is flagged, then the next observation in the sequence is processed. The sequential formulation allows sequences containing multiple change points to be easily handled; whenever a change point is detected, the change detector is simply restarted from the following observation in the sequence.

Traditionally, the tools used for both of these problems were quite different. For batch detection, the most commonly used approaches are some form of likelihood ratio testing (Hinkley and Hinkley 1970) or Bayesian inference (Stephens 1994), while the sequential detection problem uses control charts such as the CUSUM (Page 1954), Exponential Weighted Moving Averages (Roberts 1959), or sequential Bayesian methods (Chib 1998; Fearnhead and Liu 2007).

Recent years have seen the emergence of the CPM framework, which extends the use of likelihood-based batch detection methods to the problem of sequential monitoring. The original work on the CPM was presented by Hawkins *et al.* (2003) which focuses on the problem of sequentially detecting a change in the mean of a sequence of Gaussian random variables. This has since been extended in many directions to allow more complex types of changes to be detected, including those in streams where the underlying distribution is unknown (Ross *et al.* 2011; Hawkins and Zamba 2005; Zou and Tsung 2010).

The **cpm** R package contains an implementation of several different CPMs, both parametric and nonparametric, for use on univariate streams. Specifically, it implements the CPM framework using the Student-t, Bartlett, GLR, Fisher's Exact Test, Mann-Whitney, Mood, Lepage, Kolmogorov-Smirnov, and Cramer-von-Mises statistics. The first three are intended for detecting changes in sequences which are known to be Gaussian, the third is used for Bernoulli sequences, while the remainder are nonparametric and can be deployed on any stream of continuous random variables without requiring any prior knowledge of their distribution. The package is implemented in R and provides a small number of customisable high-level functions which allow the user to easily detect changes in a given stream, along with a more flexible S4 object system based representation of CPM objects which allow for greater control over the change detection procedure.

4. The change point model

The CPM extends techniques for batch detection to the sequential case. We first review the batch scenario, and then describe the sequential extension.

4.1. Batch change detection (Phase I)

In the batch scenario, there is a fixed length sequence containing the n observations $x_1, \ldots, x_n$ which may or may not contain any change point. For ease of exposition, assume that at most one change point can be present. If no change point exists, the observations are independent and identically distributed according to some distribution f_0. If a change point exists at some time τ, then the observations have a distribution f_0 prior to this point, and a distribution f_1 afterwards, where $f_0 \neq f_1$.

First consider the problem of testing whether a change occurs immediately after some specific observation k. This leads to choosing between the following two hypotheses:

$$H_0 : X_i \sim f_0(x; \theta_0), \quad i = 1, \ldots, n$$

$$H_1 : X_i \sim \begin{cases} f_0(x; \theta_0) & i = 1, 2, \ldots, k \\ f_1(x; \theta_1) & i = k+1, k+2, \ldots, n, \end{cases}$$

where θ_i represents the potentially unknown parameters of each distribution.

This is a standard problem which can be solved using a two-sample hypothesis test, where the choice of test depends on what is assumed to be known about the distribution of the observations, and the type of change which they may undergo. For example, if the observations are assumed to be Gaussian, then it would be appropriate to use a two sample Student-t test to detect a mean shift, and an F test to detect a scale shift. To avoid making such distributional assumptions, a nonparametric test can be used,, such as the Mann-Whitney test for location shifts, the Mood test for scale shifts, and the Lepage, Kolmogorov-Smirnov, and Cramer-von-Mises tests for more general changes.

After choosing a two sample test statistic $D_{k,n}$, its value can be computed and if $D_{k,n} > h_{k,n}$ for some appropriately chosen threshold $h_{k,n}$ then the null hypothesis that the two samples have identical distributions is rejected, and we conclude that a change point has occurred immediately following observation x_k.

Since it is not known in advance where the change point occurs, we do not know which value of k should be used for testing. Therefore, $D_{k,n}$ is evaluated at every value $1 < k < n$, and the maximum value is used. In other words, every possible way of splitting the data into two contiguous subsequences is considered, with a two-sample test applied at every split point. The test statistic is then:

$$D_n = \max_k \frac{D_{k,n}. - \mu_{D_{k,n}}}{\sigma_{D_{k,n}}} \quad 1 < k < n.$$

where the $D_{k,t}$statistics have been standardised to have mean 0 and variance 1 by subtracting their means $\mu_{D_{k,n}}$ and dividing by their standard deviations $\sigma_{D_{k,n}}$. The null hypothesis of no change is then rejected if $D_n > h_n$ for some appropriately chosen threshold h_n. This threshold is chosen to bound the Type 1 error rate as is standard in statistical hypothesis testing. However, this requires computing the distribution of D_n, which generally does not have an analytic finite-sample form. For some choices of the test statistics $D_{k,n}$ the asymptotic distribution of D_n can be written; for example, Hawkins (1977) derives the distribution when $D_{k,n}$ is the test statistic associated with the Student-t test, and Pettitt (1980) does similar for Mann-Whitney statistics. However, these asymptotic distributions may not be accurate when considering finite length sequences, and so numerical simulation may be required in order to estimate the distribution.

Finally, the best estimate of the change point location will be immediately following the value of k which maximised D_n, i.e.:

$$\hat{\tau} = \arg\max_k D_{k,n}. \tag{2}$$

4.2. Sequential change detection (Phase II)

The two-sample hypothesis test approach used in the batch case can be extended to sequential change detection where new observations are received over time, and multiple change points may be present. Let x_t denote the t^{th} observation that has been received, where $t \in \{1, 2, \ldots\}$.

Whenever a new observation x_t is received, the CPM approach treats $x_1, \ldots, x_t$ as being a fixed length sequence sample, computes D_t using the above batch methodology, and flags for a change if $D_t > h_t$ for some appropriately chosen threshold. If no change is detected, the next observation x_{t+1} is received, then D_{t+1} is compared to h_{t+1}, and on. The procedure therefore consists of a repeated sequence of hypothesis tests. In general, the $D_{k,n}$ statistics will have properties which allow D_{t+1} to be computed from D_t without incurring too much computational overhead; see Hawkins *et al.* (2003); Ross *et al.* (2011) for specific examples of this.

In the sequential setting, h_t is chosen so that the probability of incurring a Type 1 error is constant over time, i.e. so that under the null hypothesis of no change:

$$P(D_1 > h_1) = \alpha$$
$$P(D_t > h_t | D_{t-1} \leq h_{t-1}, \ldots, D_1 \leq h_1) = \alpha, \quad t > 1, \tag{3}$$

In this case, assuming that no change occurs, the average number of observations received before a false positive detection occurs is equal to $1/\alpha$. This quantity is referred to as the Average Run Length, or ARL_0. In general, the conditional distribution in Equation 3 is analytically intractable, and Monte-Carlo simulation is used in order to compute the required sequences of h_t values corresponding to a given choice of α. This is a computationally expensive procedure but it only needs to be carried out a single time, and the values can then be stored in a look-up table. The **cpm** package contains pre-computed sequences of thresholds which correspond to a variety of choices of α.

5. Package overview

The **cpm** package contains implementations of the CPM framework, for detecting changes in streaming sequential data (Phase II). The package supports the following CPMs:

- The *Student-t*, *Bartlett* and *GLR* statistics for detecting changes in a Gaussian sequence of random variables. The first two monitor for changes in either the mean or variance respectively, while the latter can detect changes in both.(Hawkins *et al.* 2003; Hawkins and Zamba 2005; **?**).
- The *Fisher's Exact Test* (FET) statistic for detecting a change in a sequence of Bernoulli random variables (Ross and Adams 2011a).
- The *Mann-Whitney* and *Mood* statistics for detecting location and scale changes respectively in sequences of random variables, where no assumptions are made about the distribution (Hawkins and Deng 2010; Ross *et al.* 2011).
- The *Lepage*, *Kolmogorov-Smirnov*, and *Cramer-von-Mises* statistics for detecting more general distributional changes where again no assumptions are made about the sequence distribution. (Ross and Adams 2011b).

These CPMs are implemented in C++ in order to provide fast calculations. The R interface in the package wraps around this code, and essentially has two parts, which are:

1. A set of utility functions which allow the user to easily detect changes in a sequence of observations. The core functions here are **detectChangePoint**, which tests for a single change point in a sequence of data, and **processStream** which can detect multiple

change points in a sequence, by repeatedly restarting the CPM procedure whenever a change is detected.

2. While the above functions will be sufficient for many basic change detection tasks, many streaming applications will require more nuanced control of the CPM procedure. The **cpm** package therefore provide functions which allow CPM objects to be represented as S4 objects in R, to allow greater flexibility. Here, the **makeChangePointModel** function is used to create a CPM object, and the **processObservation** function allows observations to be processed individually, with the CPM statistics being made available to the user after every observation.

All of these functions allow any of the above test statistics to be used in conjunction with CPM, and also allow a variety of different values for the ARL_0 to be specified. We will first describe how the utility functions in the package can be used to detect single change points in a sequence, and then focus on the multiple change point case. Finally, we discuss the S4 section of the package which allows greater flexibility in situations where this is required.

5.1. Detecting a single change point

The **detectChangePoint** function is used to detect a single change point in a stream of observations, and estimate its location. The use of this function can be illustrated using a simple example of a Gaussian stream which undergoes a change in mean from 0 to 1, occurring after 200 observations while the variance remains unchanged. Change detection can be performed by using (e.g.) the CPM based on the Student-t test statistic:

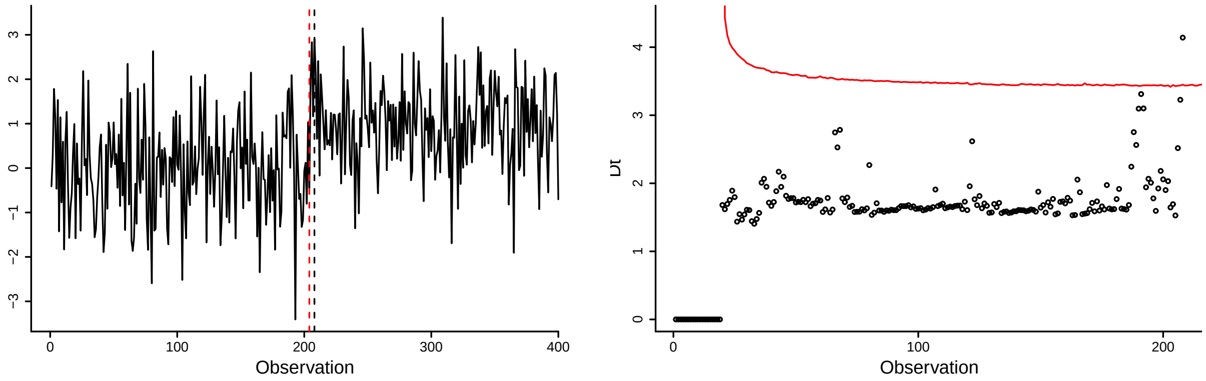
```
x <- c(rnorm(200,0,1), rnorm(200,1,1))
res <- detectChangePoint(x, cpmType = "Student", ARL0 = 500, startup = 20)

plot(x,type='l')
abline(v = res$detectionTime, lty=2)
abline(v = res$changePoint, lty=2, col='red')
```

Figure 2a shows the plot produced by this code, along with both the time at which the change was detected (which in this case occurs after the 207^{th} observation) and the estimated location of the change point, corresponding to the value of k which maximised $D_{k,207}$ as in Equation 2. In this case, it is equal to $\hat{\tau} = 202$.

The arguments taken by the **detectChangePoint** function are as follows:

- *cpmType* : determines the type of CPM to be used. The allowable values are “Student”, “Bartlett”, “GLR”, “FET”, “Mann-Whitney”, “Mood”, “Lepage”, “Kolmogorov-Smirnov”, and “Cramer-von-Mises”.
- *ARL0*: denotes which ARL_0 the CPM should have. As discussed in Section 4.2, computing the thresholds associated with a specific choice of ARL_0 can take a large amount of computation time. Therefore, the package includes precomputed values for a selection of ARL_0's, Specifically, the allowable values for the ARL0 argument are: $\{370, 500, 600, 700, ..., 1000, 2000, 3000, ..., 10000, 20000, ..., 50000\}$. If this argument is missing then no change detection will occur, and the D_t statistics will be computed and returned for the whole sequence.



(a) Observations with detected change point

(b) D_t statistics

Figure 2: The left plot shows the sequence of observations, with the detection time shown in black, and the estimated change point location shown in red. The right plot shows the values of the maximized D_t statistics up until the change is detected, with the thresholds h_t shown in red.

- *startup*: determines how many observations are used for the initialising period. If *startup* $= 20$, then no change detection will be performed until after the first 20 observations have been received. This is usually necessary since the statistical tests will have low power for small sample sizes.
- *lambda*: specifies the amount of smoothing to be used when using the Fisher's Exact Test (FET) CPM, as described in Ross and Adams (2011a). This parameter is not used with any of the other CPM types. The only allowable values for this parameter are 0.1 and 0.3; more information on this is provided later in this section.

The **detectChangePoint** function processes the sequence up until it detects a change point, after which it returns. If no change point is detected, or the ARL0 argument is not specified, the entire sequence will be processed. In either case, the following list of values is returned:

- *detectionTime*: the observation at which the change point was detected. If no change is detected, this is equal to 0.
- *changePoint*: the maximum likelihood estimate of the change point τ, defined as the value of k which maximises $D_{k,T}$, where T is the detection time
- *Ds*: The sequence of maximized test statistics, from time $t = 1$ until the detection time.
- *hs*: The sequence of h_t thresholds which correspond to the specified ARL_0 value.

The D_t statistics returned by this function can be used for diagnostics. The following code plots the values of the D_t statistics which were computed in the above example, along with the corresponding h_t thresholds:

τ	δ	CPM-Bartlett		CPM-Mood		CPM-Lepage	
$\tau = 50$	1.5	152.4	(282.0)	95.4	(190.1)	143.1	(252.0)
	2.0	16.2	(17.5)	18.0	(21.9)	26.8	(45.5)
	3.0	5.5	(3.8)	7.8	(5.3)	9.4	(6.9)
	0.5	20.3	(15.2)	38.9	(76.6)	63.4	(122.6)
	0.3	9.6	(3.7)	15.1	(5.9)	21.0	(7.7)
$\tau = 300$	1.5	30.7	(23.7)	28.0	(24.3)	35.5	(31.8)
	2.0	10.6	(7.2)	11.5	(8.3)	14.0	(10.2)
	3.0	4.6	(3.0)	6.1	(3.6)	7.5	(4.5)
	0.5	16.4	(7.5)	22.5	(27.9)	31.9	(10.7)
	0.3	9.2	(3.1)	13.1	(3.2)	18.7	(3.6)

Table 1: Average time taken to detect shifts of magnitude δ in the mean and standard deviation of a Gaussian $N(0,1)$ stream, for various change times τ. Standard deviations are given in brackets.

```
plot(res$Ds)
abline(res$hs,col='red')
```

The resulting plot is shown in Figure 2b. The first 20 values of this statistic are zero, since it is not computed during the initialisation period, defined by the *startup* argument. Then, the statistics fluctuates around a relatively flat value, until the change point occurs in which case they begin to spike. After a few observations, they cross the time varying h_t threshold, denoted by the red line, and a change is flagged.

As a slightly less trivial example, we can use the **detectChangePoint** function to compare how different CPMs perform when detecting a certain type of change. In both Hawkins and Deng (2010) and Ross *et al.* (2011), several different nonparametric CPMs are proposed which, as discussed previously, are able to maintain a specified value of the ARL_0 regardless of the true stream distribution. While this is a useful feature, the trade-off is that they will usually be slower to detect changes on any particular stream than a parametric CPM that has been designed with knowledge of the true data distribution. In the above-mentioned papers, many simulations were performed in order to measure how close the performance of the nonparametric detectors is to their parametric counterparts.

To show how such a comparison can be performed using the **cpm** package, we compare the performance of the parametric CPM which uses the Bartlett test for detecting a change in the variance of a Gaussian stream, to several different nonparametric methods. Specifically we look at the Mood CPM which is a nonparametric test for changes in scale, and the Lepage CPM which is intended to detect more general types of distributional change.

The two main factors which affect how long a change takes to be detected are the size of the change, and the number of observations which are available from the pre-change distribution. We can investigate the impact of these by considering different sets of values: specifically we consider the case where the distribution of the stream changes from $N(0,1)$ to $N(0,\delta)$ for $\delta \in \{1.5, 2.0, 3.0, 0.5, 0.3\}$ and where the change occurs at times $\tau \in \{50, 300\}$:

The following code performs the analysis:

```
cpmTypes <- c("Bartlett","Mood","Lepage")
changeMagnitudes <- c(1.5, 2.0, 3.0, 0.5, 0.3)
changeLocations <- c(50, 300)
```

```
sims <- 1000
ARL0 <- 500
startup <- 20
results <- list()

for (cpmType in cpmTypes) {
results[[cpmType]] <- matrix(numeric(length(changeMagnitudes) *
  length(changeLocations)), nrow = length(changeMagnitudes))

  for (cm in 1:length(changeMagnitudes)) {
    for (cl in 1:length(changeLocations)) {
      print(sprintf("cpm:%s magnitude::%s location:%s",
        cpmType, changeMagnitudes[cm], changeLocations[cl]))
      temp <- numeric(sims)

      for (s in 1:sims) {
        x <-c(rnorm(changeLocations[cl], 0, 1), rnorm(2000, 0,
         changeMagnitudes[cm]))

        temp[s] <-detectChangePoint(x, cpmType,
          ARL0=ARL0, startup=startup)$detectionTime
      }
      results[[cpmType]][cm,cl] <- mean(temp[temp > changeLocations[cl]]) -
        changeLocations[cl]
    }
  }
}
```

Table 1 shows the results of this analysis, when *sims* = 50000. It can be seen that, as expected, large changes are easier to detect than smaller changes, and changes which occur after the 300th observation are easier to detect than those occurring after the 50th, since the sample size is larger. In general, the parametric CPM outperforms the nonparametric CPMs, which is understandable since it incorporates knowledge that the observations are Gaussian. Interestingly, for changes with smaller magnitudes, the nonparametric CPMs actually outperform this parametric version. This surprising phenomena is discussed in more detail in both Hawkins and Deng (2010) and Ross *et al.* (2011)

As a final example, we consider the CPM which uses the FET statistic for detecting a change in a Bernoulli sequence This differs slightly from the other CPMs implemented in the package. As described in Ross and Adams (2011a), the test statistics when using the FET are highly discrete, moreso than with the other statistics considered. Therefore, a smoothing parameter λ is used to smooth the $D_{k,t}$ statistics and make them less discrete

The **detectChangePoint** function implements this by allowing a parameter to be passed to control the degree of smoothing. Because each choice of parameter value requires a different sequences of h_t thresholds to be used, we have only included threshold sequences for $\lambda = 0.1$ and $\lambda = 0.3$, which are the two values used in Ross and Adams (2011a). Note that the authors show that performance is not particularly sensitive to the choice of λ, and 0.1 can generally

θ_0	Empirical ARL_0	
	$\lambda = 0.1$	$\lambda = 0.3$
0.01	971 (862)	1039 (765)
0.05	622 (543)	638 (597)
0.10	589 (489)	529 (530)
0.20	512 (461)	516 (491)
0.30	509 (460)	509 (474)
0.40	504 (471)	507 (483)
0.50	500 (493)	500 (494)

Table 2: Empirical ARL_0 when the CPM is designed with to have an ARL_0 of 500, for several values of θ_0. Standard deviations are shown in brackets.

be used for all sequences. The option to use 0.3 is provided only for completeness.

As an illustration of how this works in practice, we will replicate Table 1 from Ross and Adams (2011a). Unlike the other test statistics, the FET CPM does not allow ARL_0 values to be achieved exactly, and the desired ARL_0 instead acts as a lower bound only. This means that the CPM is somewhat conservative, with the degree of this conservativeness depending on the value of the pre-change Bernoulli parameter θ_0. To investigate this in detail, the authors ran simulations for several values of θ_0 in order to determine the extent to which the empirical ARL_0s differs from the target value. The following code shows how this can be performed using the **cpm** package, with a target ARL_0 of 500:

```
sims <- 100
lambda <- 0.1
thetas <- c(0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.01)
ARL0s <- numeric(length(thetas))
for (i in 1:length(thetas)) {
  for (s in 1:sims) {
        x <- rbinom(10000, 1, thetas[i])
        res <- detectChangePoint(x, "FET", ARL0 = 500, startup = 20, lambda = lambda)
        ARL0s[i] <- ARL0s[i] + res$detectionTime
  }
  ARL0s[i] <- ARL0s[i] / sims
  print(sprintf("theta: %s, ARL0: %s",thetas[i], ARL0s[i]))
}
```

Table 2 shows a the resulting values, when $sims = 50000$. From this, it can be seen that the CPM is not especially conservative as long as θ_0 is greater than around 0.05. However when it drops below this value, some efficiency will be lost and it may hence take longer to detect changes than would otherwise be expected.

5.2. Streams with multiple change points

In many applications, the stream of observations may contain multiple change points. The **processStream** function can be used to detect these. The basic idea behind this function is

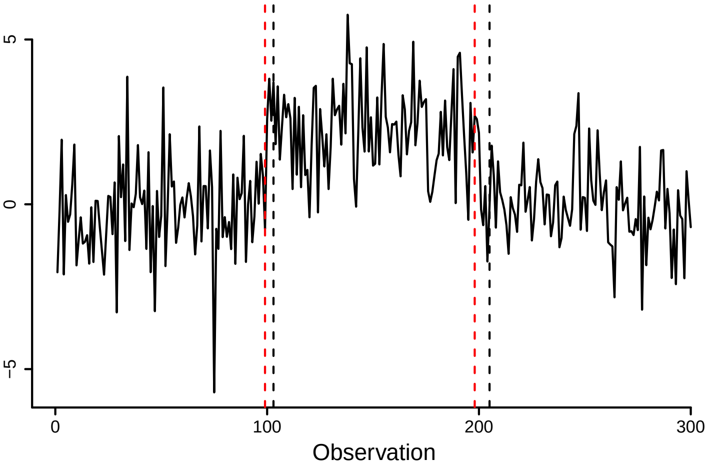


Figure 3: A sequence of Student-t(5) random variables which undergoes a change in mean after the .100th and 200th observations. The dotted black lines show the points when the change was detected, and the dotted red lines show the estimated change point locations.

to run the CPM as described above, processing each observation one-by-one. Suppose that a change is detected at time T_1, with the corresponding change point estimate being $\hat{\tau}_1$. In order to detect further change points, all the observations from before the detected change point are discarded, and a new CPM is run, beginning with the $(\hat{\tau}_1 + 1)^{th}$ observation. This is repeated until all the observations have been processed.

Figure 3 gives an illustration of this. Here, the stream consists of Student-t(5) random variables which undergo an increase in mean after the 100^{th} observation, followed by a decrease in mean after the 200^{th}. Since this is not a Gaussian stream, one of the nonparametric CPMs should be used to detect these changes.

When the CPM using the Mann-Whitney statistic is deployed on this stream, a change is detected immediately following the 105^{th} observation. The corresponding change point estimate is $\hat{\tau}_1 = 99$. After this change point has been detected, a new CPM is created and monitoring begins from the 106^{th} observation. A second change point is then detected at time $t = 205$, with the corresponding change point estimate being $\hat{\tau}_2 = 198$.

The parameters of the **processStream** function are identical to those for the **detectChange** function in the previous section. This function returns the following list of values:

- *detectionTimes*: the observation times at which the changes were detected. If multiple change points are detected then this will be a vector. If no change points are detected then it will be a vector of length zero

- *changePoints*: the maximum likelihood estimate of the change points τ_i, defined as the values of k which maximise D_{k,T_i}, where T_i is the detection time. Again, this will be a vector if multiple change points are detected, and an empty vector if no change points are detected.

Figure 3 above can hence be reproduced by the following code:

```
x <- c(rt(200,5),rt(200,5)+2,rt(200,5))
res <- processStream(x, cpmType="Mann-Whitney", ARL0=500, startup=20)
plot(x,type='l')
for (dt in res$detectionTimes) {
  abline(v=dt, lty=2)
}
for (dt in res$changePoints) {
  abline(v=dt, lty=2,col='red')
}
```

As a less trivial example of how the **processStream** function can be used for multiple change point detection, we analyse a real sequence of foreign exchange data. This example also shows how the CPM framework can be applied to sequences of observations which are not independent between the change points, by first applying a suitable transformation.

The data consists of a historical sequence of the exchange rates between the Swiss Franc (CHF) and the British Pound (GBP). The maximum value of the exchange rate was recorded at three hour intervals running from October 21^{st} 2002 to May 15^{th} 2007. In total, 9273 observations x_t were made. and we treat them as being a data stream where observations are received and processed sequentially. Although the analysis of financial data is often quite sophisticated, we provide this example to demonstrate the capabilities of our algorithm in a simplified setting.

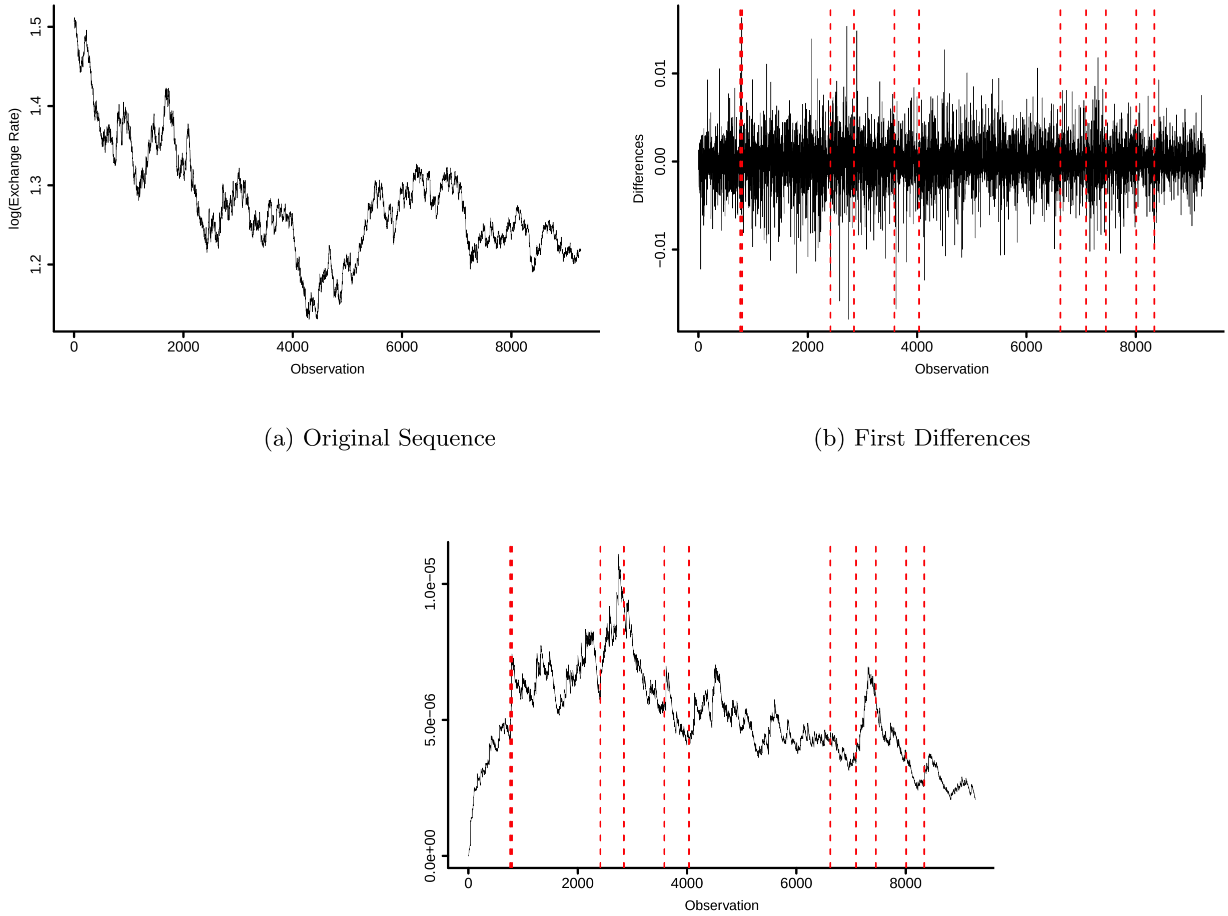
This data set is included in the **cpm** package, and can be loaded with the following command:

```
data(ForexData)

#The third column is the observed exchange rate
x <- ForexData[,3]
plot(x)
```

This produces a plot of the financial sequence, as shown in Figure 4a. From this, it is apparent that there is a high degree of autocorrelation. This is problematic for the CPM framework, which assumes that observations are independent between change points. In order to deploy the CPM, the correlation between the observations should be removed. In general, this may require modelling the time series and deploying the CPM on the residuals rather than the original data. With this financial data, we make a simple transformation and instead consider the first differences of the logarithm of the data, defined as $\Delta x_t = \log(x_t) - \log(x_{t-1})$. This transformation is typical when working with financial data, and the resulting sequence is plotted in Figure 4b, and appears stationary with mean 0. However, these first differences have very heavy tails, and this high kurtosis suggests that the data is non-Gaussian. A nonparametric change detector hence seems an appropriate tool to use for analysis.

A typical goal when analysing financial data is to detect changes in the variance (volatility) of the first differences. We use the Mood CPM for this purpose. Due to the length of the data, the ARL_0 was set to 5000 in order to avoid a large number of false alarms being generated. Because this stream is likely to contain multiple change points, the **processStream** function



(a) Original Sequence

(b) First Differences

(c) EWMA Estimate of Variance

Figure 4: The foreign exchange data, its first differences, and the EWMA of the squared first differences, all with the detected scale change points superimposed.

is used, with the CPM being restarted whenever a change is detected. The following code performs the analysis:

```
data(ForexData)

#The third column is the observed exchange rate
x <- ForexData[,3]
diffs <- numeric(length(x)-1)
for (i in 2:length(x)) {
    diffs[i] <- log(x[i]) - log(x[i-1])
}
results <- processStream(diffs, cpmType="Mood", ARL0=5000,
  startup=200)
```

This CPM detects a total of 11 change points. We have superimposed these change points on Figures 4a and 4b. It is not obvious from these plots whether the discovered change points correspond to true scale shifts, so to investigate further, we compute the exponentially weighted average of the stream variance, defined as $EWMA_t = \lambda EWMA_{t-1} + (1-\lambda)(\Delta x_t)^2$. This allows a local estimate of the variance to be formed. This EWMA is plotted iin Figure 4c, with $\lambda = 0.999$. It can be seen that the variance is undergoing gradual drift, and that the discovered change points seem to correspond to abrupt changes in the variance. The following code reproduces this EWMA analysis:

```
ewma <- numeric(length(diffs) )
ewma[1] <- diffs[1]
lambda <- 0.995
for (i in 2:length(ewma)) {
    ewma[i] <- lambda*ewma[i-1] + (1-lambda)*diffs[i]^2
}
plot(ewma,type='l')

for (i in 1:length(results$changePoints)) {
   abline(v=results$changePoints[i],lty=2)
}
```

5.3. Manipulating CPMs as S4 objects

The **detectChangePoint** and **processStream** functions provide a convenient wrapper around the internals of the CPM implementation, and will hopefully be sufficient for most problems. However in some situations more control may be required over how sequences are processed – for example, the user may wish to inspect the individual $D_{k,t}$ statistics after each observation, or the user may wish to be able to halt processing part of the way through the sequence in order to perform some further analysis.

To facilitate this, we have also implemented the CPMs as S4 objects within the **cpm** package. After a CPM object is created, it can be passed observations individually for processing. The object maintains its internal state consisting of all the observations which it has seen so far. At any point, it can be queried in order to find whether a change has been detected, or to obtain the latest set of $D_{k,t}$ statistics.

The **makeChangePointModel** function is used to create a CPM object. It takes exactly the same arguments as the **detectChangePoint** function above. After this has been created, the **processObservation** function can be used to process a sequence one observation at a time, with state being maintained after each one is processed. At any point, the CPM can be queried by using the **changeDetected** function to test whether a change has been detected, or the **getStatistics** function to gain access to the underlying $D_{k,t}$ statistics.

To illustrate this, we provide code which replicates the financial data example from the previous section. We will then explain how it works:

```
data(ForexData)
```

```
#The third column is the observed exchange rate
x <- ForexData[,3]

diffs <- numeric(length(x)-1)
for (i in 2:length(x)) {
    diffs[i] <- log(x[i]) - log(x[i-1])
}

#vectors to hold the results
detectiontimes <- numeric()
changepoints <- numeric()

#use a Lepage CPM
cpm <- makeChangePointModel(cpmType="Mood", ARL0=5000,startup=200)

i<-0
while (i < length(diffs)) {
  i <- i + 1
  #process each observation in turn
  cpm <- processObservation(cpm,diffs[i])

  #if a change has been found, log it, and reset the CPM
  if (changeDetected(cpm) == TRUE) {
    print(sprintf("Change detected at observation %d", i))
    detectiontimes <- c(detectiontimes,i)

    #the change point estimate is the maximum D_kt statistic
    Ds <- getStatistics(cpm)
    tau <- which.max(Ds)

    if (length(changepoints) > 0) {
      tau <- tau + changepoints[length(changepoints)]
    }
    changepoints <- c(changepoints,tau)

    #reset the CPM
    cpm <- cpmReset(cpm)

    #resume monitoring from the observation following the change point
    i <- tau
  }
}
```

After the CPM object has been created, the for loop iterates over the sequence one observation at a time. For each observation, the **processObservation** function is used to update the CPM. After this has been done, the **changeDetected** function is used to test whether there has been a change point. This function returns TRUE if the D_t statistic used in the CPM has exceeded the h_t threshold sequence at any point since monitoring began.

If a change point has been flagged, the **getStatistics** function is used to return the vector of $D_{k,t}$ statistics. From Equation 2, the best estimate of the change point location is the maximum value of $D_{k,t}$, and this can be found by using the built-in R
textbfwhich.max function.

After the change location has been estimated, the **cpmReset** function is used to clear the state of the CPM. When this is called, a new CPM is essentially created from scratch. Finally, the loop index i is set to the observation immediately following the detected change point, and the monitoring continues.

Note as a slight subtetly that since the CPM is completely reset after each change point, the CPM only stores the observations which occurred after the previous change point. Therefore, the following line:

```
tau <- tau + changepoints[length(changepoints)]
```

is required to convert the change point location index to an index over the whole sequence.

References

Bai JS, Perron P (1998). "Estimating and testing linear models with multiple structural changes." *Econometrica*, **66**(1), 47–78. ISSN 0012-9682.

Chib S (1998). "Estimation and comparison of multiple change-point models." *Journal of Econometrics*, **86**(2), 221–241. ISSN 0304-4076.

Efron B, Zhang NR (2011). "False discovery rates and copy number variation." *Biometrika*, **98**(2), 251–271.

Erdman C, Emerson JW (2007). "bcp: An R Package for Performing a Bayesian Analysis of Change Point Problems." *Journal of Statistical Software*, **23**(3), 1–13. URL http://www.jstatsoft.org/v23/i03/.

Fearnhead P, Liu Z (2007). "On-line inference for multiple changepoint problems." *Journal of the Royal Statistical Society Series B*, **69**(4), 589–605. ISSN 1369-7412.

Gustafsson F (2000). *Adaptive Filtering and Change Detection*. Wiley. ISBN ISBN: 978-0-470-85340-5. URL http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471492876,descCd-description.html.

Hawkins DM (1977). "Testing a Sequence of Observations For A Shift in Location." *Journal of the American Statistical Association*, **72**(357), 180–186. ISSN 0162-1459.

Hawkins DM (2001). "Fitting multiple change-point models to data." *Computuational Statistics and Data Analysis*, **37**(3), 323–341.

Hawkins DM, Deng Q (2010). "A Nonparametric Change-Point Control Chart." *Journal of Quality Technology*, **42**(2), 165–173. ISSN 0022-4065.

Hawkins DM, Qiu PH, Kang CW (2003). "The Changepoint Model for Statistical Process Control." *Journal of Quality Technology*, **35**(4), 355–366. ISSN 0022-4065.

Hawkins DM, Zamba KD (2005). "A Change-Point Model for a Shift in Variance." *Journal of Quality Technology*, **37**(1), 21–31.

Hinkley D, Hinkley E (1970). "Inference About Change-Point in a Sequence of Binomial Variables." *Biometrika*, **57**(3), 477–488. ISSN 0006-3444.

Inclan C, Tiao GC (1994). "Use of Cumulative Sums of Squares for Retrospective Detection of Changes of Variance." *Journal of the American Statistical Association*, **89**(427), 913–923. ISSN 01621459. URL `http://www.jstor.org/stable/2290916`.

Killick R, Eckley IA (2011). ***changepoint**: AnR package for changepoint analysis*. R package version 0.8-1, URL `http://CRAN.R-project.org/package=changepoint`.

Lai TL (1995). "Sequential Changepoint Detection in Quality Control and Dynamical Systems." *Journal of the Royal Statistical Society. Series B - Methodological*, **57**(4), 613–658. ISSN 00359246. URL `http://www.jstor.org/stable/2345934`.

Page ES (1954). "Continuous Inspection Schemes." *Biometrika*, **41**(1/2), 100–115. ISSN 00063444. URL `http://www.jstor.org/stable/2333009`.

Pettitt AN (1980). "A Simple Cumulative Sum Type Statistic for the Change-Point Problem With Zero-One Observations." *Biometrika*, **67**(1), 79–84. ISSN 0006-3444.

Roberts SW (1959). "Control chart tests based on geometric moving averages." *Technometrics*, **42**(1), 97–101. ISSN 0040-1706. `doi:http://dx.doi.org/10.2307/1271439`.

Ross GJ, Adams NM (2011a). "Sequential Monitoring of a Bernoulli Sequence when the Pre-change Parameter is Unknown." *Computational Statistics (under review)*.

Ross GJ, Adams NM (2011b). "Two Nonparametric Control Charts for Detecting Arbitrary Distribution Changes." *Journal of Quality Technology (In Press)*.

Ross GJ, Adams NM, Tasoulis DK, Hand DJ (2011). "A Nonparametric Change Point Model for Streaming Data." *Technometrics*, **53**(4), 379–389.

Stephens DA (1994). "Bayesian Retrospective Multiple-Changepoint Identification." *Journal of the Royal Statistical Society Series C - Applied Statistics*, **43**, 159–178.

Tartakovsky A, Rozovskii B, Blazek R, Kim; H (2006). "A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods." *IEEE Transactions on Sig*, **54**(9), 3372–3382.

Zhou C, Zou C, Zhang Y, Wang Z (2009). "Nonparametric Control Chart Based on Change-Point Model." *Statistical Papers*, **50**(1), 13–28.

Zou C, Tsung F (2010). "Likelihood Ratio-Based Distribution-Free EWMA Control Charts." *Journal of Quality Technology*, **42**(2), 174–196. ISSN 0022-4065.

Affiliation:

Gordon J. Ross
Department of Mathematics
University of Bristol
University Walk
Bristol, BS8 1TW, UK
United Kingdom E-mail: gordon.ross03@imperial.ac.uk
URL: http://gordonjross.co.uk/