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

Gordon J. Ross
Department of Statistical Science
University College London
gordon@gordonjross.co.uk

Abstract

The change point model (CPM) framework introduced in Hawkins, Qiu, and Kang (2003) and Hawkins and Zamba (2005a) provides an effective and computationally efficient method for detecting multiple mean or variance change points in sequences of Gaussian random variables, when no prior 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, Tasoulis, and Adams (2011), Ross and Adams (2012) 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 (2011) and Ross (2014) which allows change detection in streams of Bernoulli and Exponential random variables respectively, 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 in both batch (Phase I) and sequential (Phase II) settings, where the sequences may contain either a single or multiple change points.

Keywords: —!!!—at least one keyword is required—!!!—.

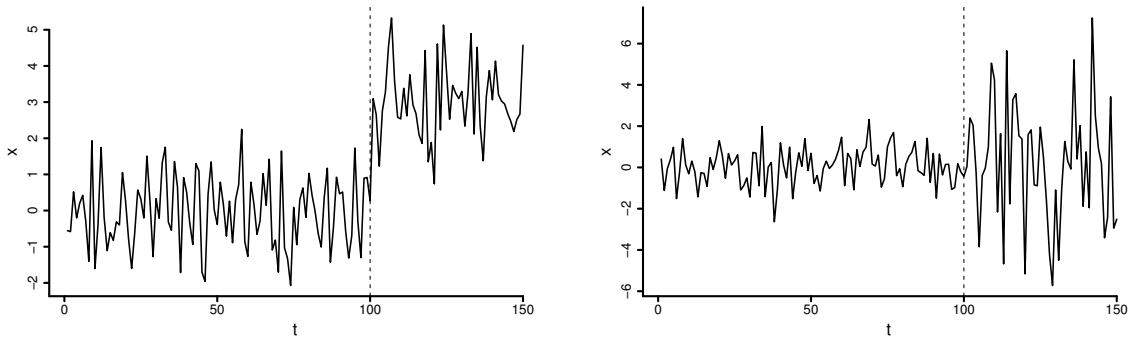
1. Introduction

Many statistical problems require change points to be identified in sequences of data. In the usual setting, a sequence of observations $x_1, x_2, \ldots$ are 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 and identically distributed between every pair of change points, so that the distribution of the sequence can be written as:

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

where the F_is represent the distribution in each segment. Some simple examples are given in Figure 1, which shows two sequences of Gaussian random variables that undergo changes in their mean and variance respectively.

Although requiring independence between change-points seems a restrictive assumption, this



(a) Mean Change. (b) Variance Change.

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

is not the case since dependence can typically be handled by first modeling any underlying dynamics or autocorrelation, and then performing change detection on either the model residuals or 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 apart from the example in Section 4.2, it is assumed that such modeling has already been carried out, giving 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 2005a; Zhou, Zou, Zhang, and Wang 2009; Hawkins and Deng 2010; Ross *et al.* 2011; Ross and Adams 2011, 2012).

The purpose of this paper is to describe the **cpm** package, which is written in R (R Core Team 2013) and 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), **strucchange** (Zeileis, Leisch, Hornik, and Kleiber 2002) and **changepoint** (Killick and Eckley 2011), the **cpm** allows sequential Phase II analysis when the parameters and perhaps distributional form of the observations are unknown.

Section 2 gives a more detailed overview of the sequential change detection problem. Section 3 provides a general overview of the CPM framework. Finally, the **cpm** package is introduced in Section 4 and an overview of its capabilities is given, along with many examples of usage. This package can be obtained from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=cpm`. To cite either the package or this document, please

use the citation "G. J. Ross - Parametric and Nonparametric Sequential Change Detection in R: The cpm Package, Journal of Statistical Software, 2015, forthcoming".

2. 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 genomic 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 (Ross 2012).

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 (Hawkins and Zamba 2005a; Zou and Tsung 2010; Ross *et al.* 2011).

The R package **cpm** contains an implementation of several different CPMs, both parametric and nonparametric, for use on univariate streams in both the Phase I and Phase II setting. Specifically, it implements the CPM framework using the Student-t, Bartlett, GLR, Fisher's exact test, Exponential, 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 fourth is used for Bernoulli sequences, the fifth for Exponential 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 customizable 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.

3. The CPM

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

3.1. 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 a 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:

$$
\begin{aligned}
H_0 &: X_i \sim F_0(x; \theta_0), \quad i = 1, \ldots, n \\
H_1 &: X_i \sim \begin{cases} F_0(x; \theta_0) & i = 1, 2, \ldots, k \\ F_1(x; \theta_1) & i = k+1, k+2, \ldots, n, \end{cases}
\end{aligned}
$$

where θ_i represent 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 statistic 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 $D_{k,n}$ exceeds 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:

$$D_n = \max_{k=2,\ldots,n-1} \left| \frac{D_{k,n} - \mu_{D_{k,n}}}{\sigma_{D_{k,n}}} \right|,$$

where the $D_{k,n}$ statistics have been standardized to have mean 0 and variance 1 by subtracting their means $\mu_{D_{k,n}}$ and dividing by their standard deviations $\sigma_{D_{k,n}}$. Note that the modulus should be taken to allow for two-sided change detection when using a statistic where both very low and very high values indicate a difference between the samples (e.g., being able to detect both increases and decreases in the mean and/or variance).

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. Suppose α is an acceptable level for the proportion of false positives, i.e., the probability of falsely declaring that a change has occurred if in fact no change has occurred. Then, h_n should be chosen as the upper α quantile of the distribution of D_n under the null hypothesis.

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, Pettitt (1979) does similar for Mann-Whitney statistics, and a general procedure for asymptotically bounding D_n for other classes of test statistic is given by Worsley (1982). 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. The latter approach is used in the **cpm** package.

Finally, the best estimate of the change point location will be immediately following the value of k which maximized D_n:

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

3.2. Sequential change detection (Phase II)

The two-sample hypothesis testing 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 tth 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 and computes D_t using the above batch methodology, where we are using the notation D_t rather than D_n to highlight the sequential nature of the procedure. A change is then flagged 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 computed and compared to h_{t+1}, and so 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) and 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, so that under the null hypothesis of no change:

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

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.

4. Package overview

The **cpm** package contains implementations of the CPM framework, for detecting changes in either a batch of data containing at most one change point (Phase I), or a sequential context with data which may contain multiple change points (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 2005a,b).

- The *Exponential* statistic for detecting a parameter change in a sequence of Exponentially random variables (Ross 2014).

- The *GLRAdjusted* and *ExponentialAdjusted* statistics which are identical to the GLR and Exponential statistics, except for using the finite sample correction described in Ross (2014) which can lead to more powerful change detection.

- The *Fisher's exact test* (FET) statistic for detecting a change in a sequence of Bernoulli random variables (Ross and Adams 2011).

- 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 2012).

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

1. A set of utility functions allowing the user to easily detect changes in a sequence of observations. The core functions here are `detectChangePointBatch` and `detectChangePoint` which perform Phase I and Phase II detection respectively for sequences containing at most one change point, and `processStream` which can detect multiple change points in a sequence, by repeatedly restarting the Phase II CPM procedure whenever a change is detected.

2. While the above functions will be sufficient for many basic change detection tasks, many streaming Phase II applications will require more nuanced control of the CPM procedure. The **cpm** package therefore provides functions which allow CPM objects to be represented as S4 objects in R and thus allowing for greater flexibility. Here, the function `makeChangePointModel` is used to create a CPM object and the function `processObservation` 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 any CPM, and also allow a variety of different values for the ARL_0 to be specified. We will first describe in Section 4.1 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 in Section 4.2. Finally, we discuss the S4 section of the package in Section 4.3 which allows greater flexibility in situations where this is required.

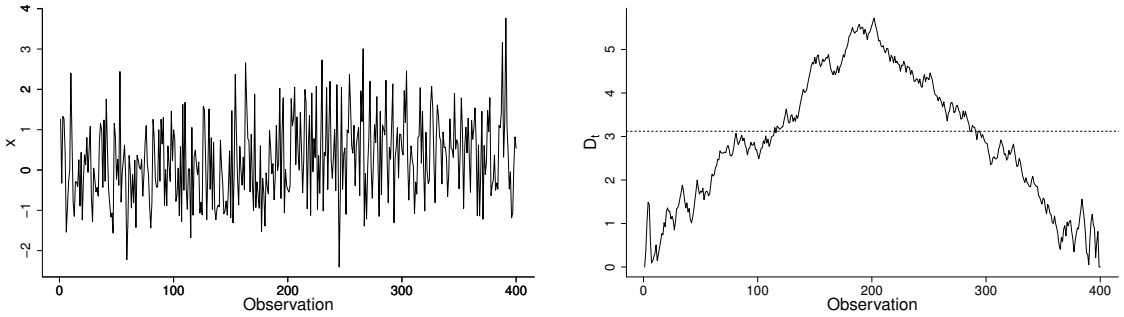
4.1. Detecting a single change point

We begin by describing how to perform Phase I analysis, and then focus on Phase II.

Phase I

The `detectChangePointBatch` function is used to test for a single change point in a sequence of observations, and estimate its location. In keeping with Phase I analysis, the data is processed in one batch rather than sequentially. 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 0.5, occurring after 200 observations while the variance remains unchanged. Parametric and nonparametric change detection can be performed by using the CPM based on the Student-t and Mann-Whitney test statistics respectively:

```
R> set.seed(0)
R> x <- c(rnorm(200, 0, 1), rnorm(200, 0.5, 1))
R> resultsStudent <- detectChangePointBatch(x, cpmType = "Student",
+    alpha = 0.05)
R> resultsMW <- detectChangePointBatch(x, cpmType = "Mann-Whitney",
+    alpha = 0.05)
R> plot(x, type = "l")
R> if (resultsStudent$changeDetected)
R>   abline(v = resultsStudent$changePoint, lty = 2)
```



(a) Observations with detected change point.

(b) $D_{k,n}$ statistics.

Figure 2: The left plot shows the sequence of observations, with the estimated change point location found using both the Student-t and Mann-Whitney CPM as a dashed line. The right plot shows the associated $D_{k,n}$ with the h_n threshold shown as a horizontal dashed line.

```
R> if (resultsMW$changeDetected)
R>   abline(v = resultsMW$changePoint, lty = 2)
```

Figure 2a shows the plot produced by this code, along with both the best estimate of the change point, corresponding to the value of k which maximized $D_{k,n}$ as in Equation 2. In this case, both the Student-t and Mann-Whitney CPMs estimate the change point to be at the same location, $k = 202$.

The arguments taken by the `detectChangePointBatch` function are as follows:

- `x`: the sequence to be analyzed, represented as a standard numeric vector.
- `cpmType`: determines the type of CPM to be used. The allowable values are `"Student"`, `"Bartlett"`, `"GLR"`, `"GLRAdjusted"`, `"Exponential"`, `"ExponentialAdjusted"`, `"FET"`, `"Mann-Whitney"`, `"Mood"`, `"Lepage"`, `"Kolmogorov-Smirnov"`, `"Cramer-von-Mises"`. With the exception of FET, these CPMs are all implemented in their two sided forms, and are able to detect both increases and decreases in the parameters monitored.
- `alpha`: the null hypothesis of no change is rejected if $D_n > h_n$ where n is the length of the sequence and h_n is the upper α (alpha) percentile of the test statistic distribution. Because computing the values of h_n associated with each value of α is a laborious task, the package includes a function `getBatchThreshold` which returns the threshold associated with a particular choice of α and n. This function is called automatically whenever `detectChangePointBatch` is called, so the user only needs to provide the desired value of α. The allowable values for this argument are 0.05, 0.01, 0.005, 0.001. If a different value is required then the user will need to compute it manually.
- `lambda`: specifies the amount of smoothing to be used when using the FET CPM, as described in Ross and Adams (2011). 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 `detectChangePointBatch` function returns the following arguments:

- `changeDetected`: `TRUE` if D_n exceeds the value of h_n associated with α, otherwise `FALSE`.
- `changePoint`: assuming a change was detected, this stores the most likely location of the change point, defined as the value of k which maximized $D_{k,n}$. If no change is detected, this is set to 0.
- `Ds`: the sequence of test statistics $D_{k,n}$.
- `threshold`: the value of h_n which corresponds to the specified α.

The $D_{k,n}$ statistics returned by this function can be used for diagnostics. The following code plots the values of the $D_{k,n}$ statistics which were computed in the above example, along with the h_n threshold corresponding to $\alpha = 0.05$.

```
R> plot(resultsMW$Ds, xlab = "Observation", ylab = expression(D[t]),
+    pch = 19)
R> abline(h = resultsMW$threshold, lty = 2)
```

The resulting plot is shown in Figure 2b. It can be seen that the test statistics start to peak around the 200th observation where the change occurs, as expected.

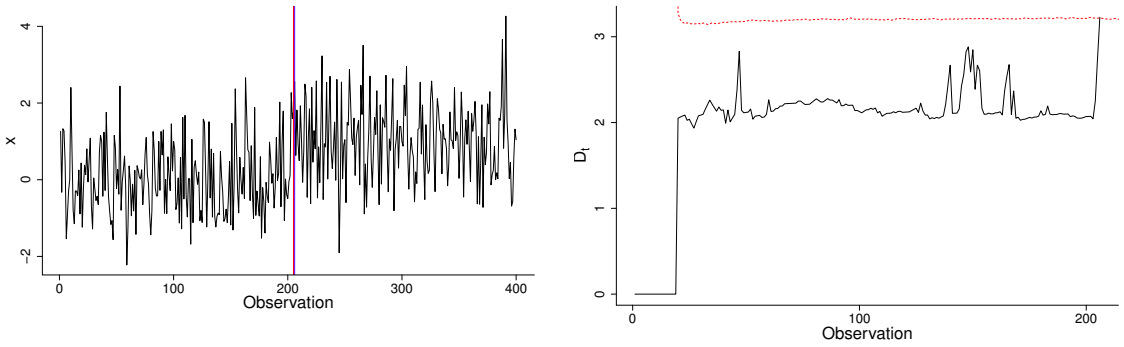
Phase II

The `detectChangePoint` function allows Phase II analysis to be performed, where the observations are processed sequentially rather than in batch. In this case the goal is usually to detect the change point as soon after it occurs as possible. The arguments to this function are similar to those of the `detectChangeBatch` function in the previous section and are:

- `cpmType`: as before.
- `ARL0`: denotes which ARL_0 the CPM should have. As discussed in Section 3.2, 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 argument `ARL0` are: $\{370, 500, 600, 700, \ldots, 1000, 2000, 3000, \ldots, 10000, 20000, \ldots, 50000\}$. 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.
- `startup`: determines how many observations are used for the initialization 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`: as before.

The values returned by the function are:

- `changeDetected`: `TRUE` if any value of D_t exceeds the corresponding threshold h_t, otherwise `FALSE`.



(a) Observations with detected change point.

(b) D_t statistics.

Figure 3: The left plot shows the sequence of observations, with the point where the change was detected using both the Student-t and Mann-Whitney CPMs as a red and blue line respectively. The right plot shows the values of the D_t statistics from the Mann-Whitney CPM up until the change is detected, with the thresholds h_t shown as a red line.

- `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 for which $D_{k,n}$ is maximum when $n = T$ and T is the detection time. If no change is detected, this is equal to 0.
- `Ds`: the sequence of maximized test statistics, from time $t = 1$ until the detection time.
- `thresholds`: the sequence of h_t thresholds which correspond to the specified ARL_0 value.

As an example of this function being used, we repeat the analysis from the previous section except for considering now a Phase II setting, comparing the times at which the Student-t and Mann-Whitney CPMs detect that a change has occurred in a sequence of Gaussian random variables:

```
R> set.seed(0)
R> x <- c(rnorm(200, 0, 1), rnorm(200, 1, 1))
R> resultsStudent <- detectChangePoint(x, cpmType = "Student", ARL0 = 500)
R> resultsMW <- detectChangePoint(x, cpmType = "Mann-Whitney", ARL0 = 500)
R> plot(x, type = "l")
R> if (resultsStudent$changeDetected)
R>   abline(v = resultsStudent$detectionTime)
R> if (resultsMW$changeDetected)
R>   abline(v = resultsMW$detectionTime, lty = 2)
```

Figure 3a shows the sequence of observations along with the time at which the change was detected by both CPMs. In this case the Student-t version detects it faster ($t = 205$ compared

to $t = 206$), which is expected since it is using extra information about the parametric form of the sequence. Again, the D_t statistics returned by this function can be used for diagnostics: note that while in the batch case the individual statistics $D_{k,n}$ are returned, in the sequential case it is the maximized D_t statistics which are returned (where D_t is defined as in Section 3.2). These statistics are plotted in Figure 3b. The first 20 values of this statistic are zero, since it is not computed during the initialization period, defined by the `startup` argument. Then, the statistics fluctuates around a relatively flat value, until the change point occurs after which they begin to spike. After a few observations, they cross the time varying h_t thresholds, denoted by the gray 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, as was the case in the previous example where the Mann-Whitney CPM was slower to detect the Gaussian mean shift than the Student-t CPM. 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.

For a fixed value of the ARL_0, 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 Appendix contains the R code used to perform this analysis and Table 1 shows the results, averaged over 100 simulations (note that we have kept the number of simulations low for computational reasons; more accurate results will be obtained with a higher number of simulations, but the qualitative results in the table do not change). 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 investigate the impact that different choices of the ARL_0 have on performance. Choosing an appropriate value for the ARL_0 depends on the particular application being considered; low values typically result in less delay until change points are detected, at

τ	δ	Bartlett		Mood		Lepage	
	1.5	138.4	(193.9)	88.2	(138.5)	148.5	(271.2)
	2.0	16.2	(15.2)	18.9	(17.0)	32.5	(79.8)
$\tau = 50$	3.0	5.2	(3.4)	7.4	(4.7)	8.4	(5.8)
	0.5	19.0	(12.9)	35.4	(27.8)	52.4	(37.5)
	0.3	8.8	(3.0)	13.7	(5.0)	18.6	(6.1)
	1.5	25.5	(19.2)	26.7	(18.0)	25.2	(17.2)
	2.0	11.5	(7.0)	12.6	(9.4)	17.0	(13.3)
$\tau = 300$	3.0	4.6	(3.0)	6.0	(3.5)	7.8	(4.3)
	0.5	14.9	(6.9)	24.0	(10.9)	31.4	(8.6)
	0.3	8.0	(3.0)	11.9	(2.2)	18.0	(2.8)

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.

ARL_0	Delay	False Positives
100	20.04	0.55
500	47.29	0.17
1000	50.07	0.06

Table 2: Average time to detect a change from 0.4 to 0.6 in a Bernoulli parameter occurring after 100 observations, for various choices of ARL_0. The proportion of times a false positive was signaled is also given.

the cost of incurring a greater number of false positives. We illustrate this by using the FET CPM to detect a change of a fixed magnitude in a Bernoulli sequence, for different choices of the ARL_0.

Note that the FET CPM differs slightly from the other CPMs implemented in the package. As described in Ross and Adams (2011), the test statistics when using the FET are highly discrete, even more so 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. Both the `detectChangePointBatch` and `detectChangePoint` functions implement this by allowing a parameter `lambda` 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 (2011). Note that the authors show that performance is not particularly sensitive to the choice of λ, and 0.1 can generally be used for all sequences. The option to use 0.3 is provided only for completeness. Finally, for reasons discussed in Ross and Adams (2011) the test statistic discreteness means that the FET CPM is conservative when the pre-change proportion θ_0 is less than 0.1, and so the achieved ARL_0 may exceed the desired ARL_0 for small values of θ_0.

The Appendix contains code that can be used to calculate the performance FET CPM when detecting a change from $\theta_0 = 0.4$ to $\theta_1 = 0.6$, occurring after 100 observations, again using 100 simulations. Table 2 shows the average delay until the change is detected for several choices of the ARL_0, along with the proportion of times a change was flagged before the change occurs

at the 100th, corresponding to a false positive. As expected, a lower value of the ARL_0 results in faster change detection, at the cost of higher false positives. Again in practice the user should typically decide what sort of false positive rate is acceptable, and choose the ARL_0 appropriately.

4.2. Sequences containing 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 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 4 gives an illustration of this. Here, the stream consists of Student-$t(5)$ random variables which undergo an increase in mean after the 50th observation, followed by a decrease in mean after the 100th. 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 53rd observation. The corresponding change point estimate is $\hat{\tau}_1 = 49$. After this change point has been detected, a new CPM is created and monitoring begins from the $49 + 1 = 50$th observation. A second change point is then detected at time $t = 105$, with the corresponding change point estimate being $\hat{\tau}_2 = 100$.

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 maximize 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 4 above can hence be reproduced by the following code:

```
R> set.seed(0)
R> x <- c(rt(50, 5), rt(50, 5) + 3, rt(50, 5))
R> res <- processStream(x, cpmType = "Mann-Whitney", ARL0 = 500,
+    startup = 20)
R> plot(x, type = "l")
R> abline(v = res$detectionTimes)
R> abline(v = res$changePoints, lty = 2)
```

As a less trivial example of how the `processStream` function can be used for multiple change point detection, we analyze a real sequence of foreign exchange data. This example also

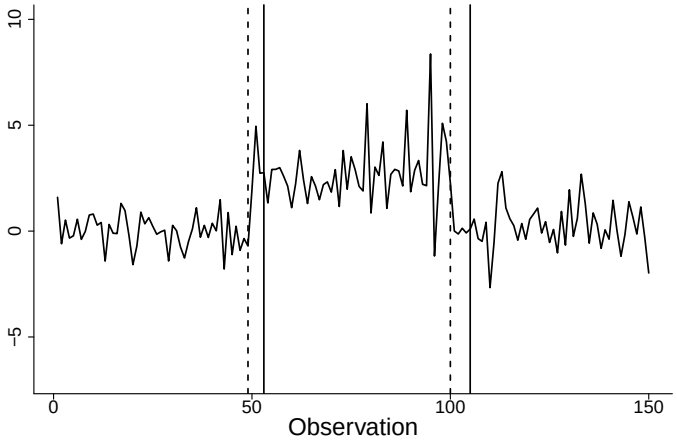


Figure 4: A sequence of Student-$t(5)$ random variables which undergoes a change in mean after the 50th and 100th observations. The solid black lines show the points when the change was detected, and the dashed black lines show the estimated change point locations.

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 21st 2002 to May 15th 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.

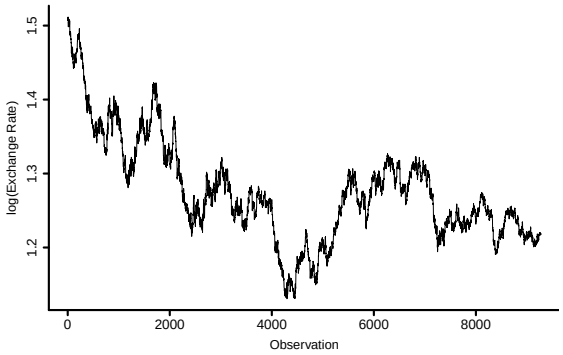
This data set is included in the **cpm** package, and can be loaded with the following command:

```
R> data("ForexData", package = "cpm")
```

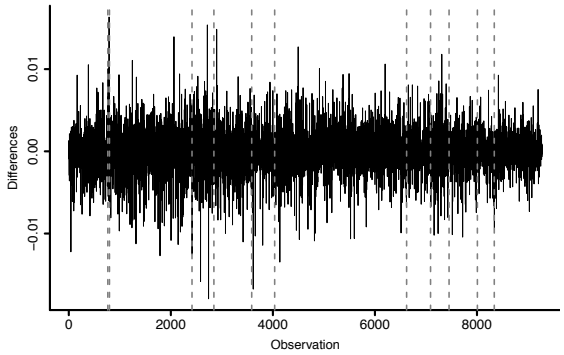
The third column of this matrix contains the logarithm of the maximum value of the exchange rate for each period. This can be selected using:

```
R> x <- ForexData[, 3]
R> plot(x)
```

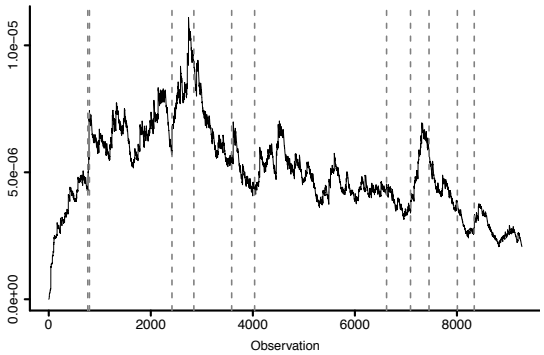
This produces a plot of the financial sequence, as shown in Figure 5a. From this, it is apparent that there is a high degree of autocorrelation. This is problematic for the CPM framework, which assumes that observations are independent between change points. In order to deploy the CPM, the correlation between the observations should be removed. In general, this may require modeling 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 $\Delta x_t = x_t - x_{t-1}$. This transformation is typical when working with financial data, and the resulting sequence is plotted in Figure 5b, and appears stationary with mean 0. However, these first differences



(a) Original Sequence.



(b) First Differences.



(c) EWMA Estimate of Variance.

Figure 5: The foreign exchange data, its first differences, and the EWMA of the squared first differences, all with the detected scale change points superimposed.

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

```
R> results <- processStream(diffs, cpmType = "Mood", ARL0 = 5000,
+    startup = 20)
```

This CPM detects a total of 11 change points. We have superimposed these change points on Figure 5b. It is not obvious from this plot 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 in Figure 5c, with $\lambda = 0.995$. 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:

```
R> ewma <- numeric(length(diffs))
R> ewma[1] <- diffs[1]^2
R> lambda <- 0.995
R> for (i in 2:length(ewma))
+    ewma[i] <- lambda * ewma[i - 1] + (1 - lambda) * diffs[i]^2
R> plot(ewma, type = "l")
R> abline(v = results$changePoints, lty = 2)
```

4.3. 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. First, the data is loaded and two arrays are created to contain the detection times and change points:

```
R> detectiontimes <- numeric()
R> changepoints <- numeric()
```

Next, a Lepage CPM is created, and the observations are iterated over one by one:

```
R> cpm <- makeChangePointModel(cpmType = "Mood", ARL0 = 5000, startup = 20)
R> i <- 0
R> while (i < length(diffs)) {
+    i <- i + 1
```

```
+    cpm <- processObservation(cpm, diffs[i])
+    if (changeDetected(cpm) == TRUE) {
+      cat(sprintf("Change detected at observation %d\n", i))
+      detectiontimes <- c(detectiontimes, i)
+      Ds <- getStatistics(cpm)
+      tau <- which.max(Ds)
+      if (length(changepoints) > 0)
+        tau <- tau + changepoints[length(changepoints)]
+      changepoints <- c(changepoints, tau)
+      cpm <- cpmReset(cpm)
+      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 $D_{k,n}$ statistics where n is equal to the index of the last observation processed (i.e. if 100 observations have been processed, then `getStatistics` will return the values $D_{1,1}, \ldots, D_{1,100}$). From Equation 2, the best estimate of the change point location is the maximum value of $D_{k,n}$, and this can be found by using the built-in R function `which.max`.

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.

The estimated change points are as follows:

```
R> changepoints

[1]  765  797 2415 2844 3585 4035 6621 7090 7453 8008 8339
```

which match the values returned by the `processStream` function in the previous section. Note as a slight subtlety 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:

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

was included in the above loop to convert the change point location index to an index over the whole sequence.

5. Additional Code

The following code produces the results shown in Table 1 which compares the performance of the Bartlett and Mood CPMs:

```
R> set.seed(0)
R> cpmTypes <- c("Bartlett","Mood","Lepage")
R> changeMagnitudes <- c(1.5, 2.0, 3.0, 0.5, 0.3)
R> changeLocations <- c(50, 300)
R> sims <- 100
R> ARL0 <- 500
R> startup <- 20
R> results <- list()

R> 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]
+     }
+   }
+ }
```

The following code produces the results shown in Table 2 which compares how different choices of the ARL_0 affect the detection delay:

```
R> set.seed(0)
R> theta0 <- 0.4
R> theta1 <- 0.6
R> ARL0s <- c(100,500,1000)
R> startup <- 20
R> changePoint <- 100
R> sims <- 100

R> results <- matrix(numeric(length(ARL0s)*sims),ncol=sims)
R> for (i in 1:length(ARL0s)) {
+  for (s in 1:sims) {
+    x <-c(rbinom(changePoint,1,theta0), rbinom(500,1,theta1))
```

```
+    results[i,s] <- detectChangePoint(x, "FET",
+    ARL0=ARL0s[i], startup=startup,lambda=0.1)$detectionTime
+  }
}

#average delays
R> apply(results,1,function(z) {mean(z[z>=changePoint])-changePoint})

#proportion of false positives
R> apply(results,1,function(z) {length(z[z<changePoint])/sims})

#output results as a table
R> library(xtable) #used to produce LaTeX markup
R> M <- cbind(ARL0s,
+  apply(results,1,function(z) {mean(z[z>=changePoint])-changePoint}),
+  apply(results,1,function(z) {length(z[z<changePoint])/sims}))
R> colnames(M) <- c("ARL0", "Delay", "False Positives")
R> rownames(M) <- NULL
R> xtable(M,label=NULL)
```

References

Chib S (1998). "Estimation and Comparison of Multiple Change-Point Models." *Journal of Econometrics*, **86**(2), 221–241.

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 B*, **69**(4), 589–605.

Gustafsson F (2000). *Adaptive Filtering and Change Detection*. John Wiley & Sons, Hoboken, NJ.

Hawkins DM (1977). "Testing a Sequence of Observations for a Shift in Location." *Journal of the American Statistical Association*, **72**(357), 180–186.

Hawkins DM (2001). "Fitting Multiple Change-Point Models to Data." *Computational 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.

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

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

Hawkins DM, Zamba KD (2005b). "Statistical Process Control for Shifts in Mean or Variance Using a Changepoint Formulation." *Technometrics*, **47**(2), 164–173.

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

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.

Killick R, Eckley IA (2011). ***changepoint**: An R Package for Changepoint Analysis*. R package version 0.6, 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 B*, **57**(4), 613–658.

Page ES (1954). "Continuous Inspection Schemes." *Biometrika*, **41**(1/2), 100–115.

Pettitt AN (1979). "A Non-Parametric Approach to the Change-Point Problem." *Journal of the Royal Statistical Society C*, **28**(2), 126–135.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org.

Roberts SW (1959). "Control Chart Tests Based on Geometric Moving Averages." *Technometrics*, **42**(1), 97–101.

Ross GJ (2012). "Modelling Financial Volatility in the Presence of Abrupt Changes." *Physica A: Statistical Mechanics and its Applications*, **392**(2), 350–360.

Ross GJ (2014). "Sequential Change Detection in the Presence of Unknown Parameters." *Statistics and Computing*, **14**(6), 1017–1030.

Ross GJ, Adams NM (2011). "Sequential Monitoring of a Bernoulli Sequence when the Pre-change Parameter is Unknown." *Computational Statistics*, **28**(2), 463–479.

Ross GJ, Adams NM (2012). "Two Nonparametric Control Charts for Detecting Arbitrary Distribution Changes." *Journal of Quality Technology*, **44**(12), 102–116.

Ross GJ, Tasoulis DK, Adams NM (2011). "Nonparametric Monitoring of Data Streams for Changes in Location and Scale." *Technometrics*, **53**(4), 379–389.

Stephens DA (1994). "Bayesian Retrospective Multiple-Changepoint Identification." *Journal of the Royal Statistical Society C*, **43**(1), 159–178.

Tartakovsky AG, Rozovskii BL, Blazek RB, 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 Signal Processing*, **54**(9), 3372–3382.

Worsley KJ (1982). "An Improved Bonferroni Inequality and Applications." *Biometrika*, **69**(2), 297–302.

Zeileis A, Leisch F, Hornik K, Kleiber C (2002). "strucchange: An R Package for Testing for Structural Change in Linear Regression Models." *Journal of Statistical Software*, **7**(2), 1–38. URL http://www.jstatsoft.org/v07/i02.

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.

Affiliation:

Gordon J. Ross
Department of Statistical Science
University College London
1-19 Torrington Place
London, WC1E 7JE, UK
United Kingdom
E-mail: gordon@gordonjross.co.uk
URL: http://gordonjross.co.uk/