Detecting Abrupt Changes in Big Data

Kaylea Haynes, B.Sc.(Hons.), M.Res

Submitted for the degree of Doctor of Philosophy at Lancaster University.

March 2017

Detecting Abrupt Changes in Big Data

Kaylea Haynes, B.Sc.(Hons.), M.Res

Submitted for the degree of Doctor of Philosophy at Lancaster University.

September 2016

Abstract

This thesis looks at developing methods for changepoint detection that can be used in the realm of Big Data. In particular we look at developing methods that can be scaled to the volume of data, now readily collected and stored, and are also versatile to the different varieties of data.

A well established approach to detect changes uses penalised optimisation where the choice of the penalty has a huge impact on the performance of the method. In the first part of this thesis we propose an algorithm, CROPS (Changepoints over a Range of PenaltieS), which finds the optimal solutions for a range of penalties instead of only specifying one penalty.

The second part of this thesis looks at the choice of cost function used in the optimisation. In particular we develop a computationally efficient method, which uses a nonparametric cost function, allowing for changes to be detected in a larger variety of data-sets. This nonparametric approach uses the empirical cumulative distribution of the data and thus does not require any assumptions to be made on distributional parameters.

The third part of this thesis looks at ways to parallelise detection methods in order to use multi-core computers and thus allowing for changes to be detected in much larger data-sets than they could be previously. We look at different ways to split the data across multiple cores and then merge the results to try to conserve as much of the accuracy that we had when we only used one core.

Acknowledgements

There are many people I need to thank whom this PhD would not have been possible without. First and foremost, I would like to thank my supervisors Professors Paul Fearnhead and Idris Eckley for all their help, support and patience throughout this PhD. I have learnt many lessons during this PhD from both Paul and Idris and for that I am very grateful. Thanks also to Dr Rebecca Killick for involving me in the changepoint R package and for all the insightful discussions over the past few years.

I am very grateful for the financial support provided by the EPSRC and DSTL. I would like to thank my industrial supervisor Ralph Mansson for his helpful conversations and for organising my trip to DSTL which gave me insight into working in industry. I also need to thank Vincent Brault for providing access to the Hi-C data used in Chapter 3.

I would like to thank all the staff at the STOR-i Centre for Doctoral Training for providing a stimulating and enjoyable research environment, as well providing many training opportunities. A special thanks to Professor Jonathan Tawn for his continuous words of support and guidance.

I would also like to thank all of the support staff at STOR-i and Lancaster University. Especially to Cyrus Gaviri and Dave Sole who have provided me with lots of IT help and have helped sort out even my more awkward of computer problems.

My PhD experience would not have been what it is today without all the friends I have made along the way. A special thanks to the surviving students in my cohort: Christian, Ivar, Lawrence, Lisa and Monika, who have been on this journey with

me from the start. Their kindness, laughter and friendship has really helped make this experience enjoyable and helped me through the less fun parts. Thanks also to the changepoint reading group and the machine learning reading group for all of the interesting discussions but most importantly for all the coffee and cake!

On a more personal note I am very grateful to my parents and family for supporting me over the years though I'm sure they've been counting down the days to when I'm no longer living the student life.

Lastly a huge thanks to Rob for all our changepoint chat but also for his love and support during the write up phase of this thesis.

Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

Chapter 3 has been accepted for publication as Haynes, K., Eckley, I.A., and Fearnhead, P. (2016). Computationally efficient changepoint detection for a range of penalties. *Journal of Computational and Graphical Statistics*.

Chapter 4 has been accepted for publication as Haynes, K., Fearnhead, P., and Eckley, I.A. (2016). A computationally efficient nonparametric approach for changepoint detection. *Statistics and Computing*.

Kaylea Haynes

Contents

List of Figures

List of Tables

Chapter 1

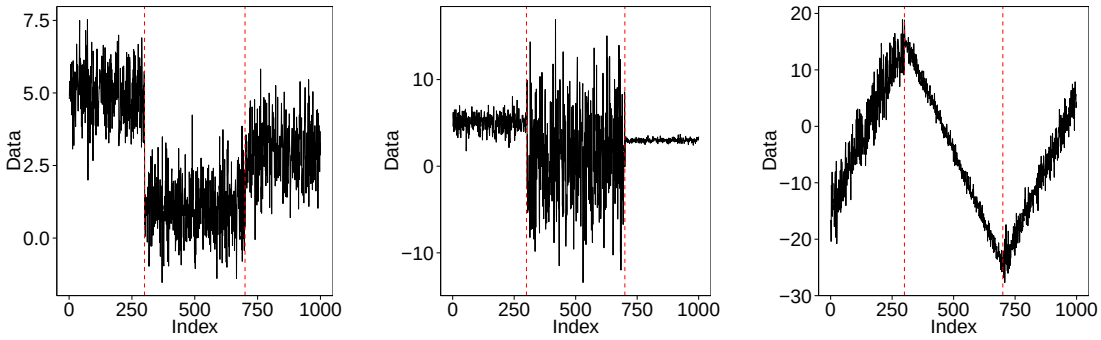
Introduction

High resolution data sensors are common-place in the devices which we use in our day to day lives. For example, mobile phones contain many sensors for measuring motion, orientation and environmental conditions (Android, 2016). Consequently we are now able to record and store more data than ever before. This has resulted in a resurgence of interest in a number of different inference areas, not least of which is changepoint analysis.

A *changepoint* (sometimes referred to as a breakpoint) is a point in a data-series, for example a time-point or a position along a chromosome, where there has been a change in one or more of the statistical properties. Figure 1.1 shows example data-sets with (a) changes in mean, (b) changes in mean and variance and (c) changes in trend and variance. Knowledge of these changepoints is often invaluable for effective modelling and forecasting.

In this thesis we look at developing methods for changepoint detection that can be applied in the *Big Data* revolution. We will start by reviewing some of the vast literature of changepoint detection in Chapter 2. In particular we will focus our attention on methods for detecting multiple changes in a univariate, offline setting since this is the scenario of main interest throughout this thesis.

A popular approach for changepoint detection is to use penalised optimisation



(a) (b) (c)

Figure 1.1: Examples of data-series with changes in one or more statistical property: (a) changes in mean, (b) changes in mean and variance and (c) changes in trend and variance.

which requires a choice of a penalty. Mis-specifying this penalty can have a detrimental effect in the performance of the changepoint detection method. In Chapter 3 we propose a new algorithm, CROPS (Changepoints over a Range Of PenaltieS), which finds the optimal solution over a range of penalties in a continuous range. We apply CROPS to detect genomic regions that interact through the folding and 3-D structure of a chromosome and show how we can choose the best segmentation once we have recovered the segmentations for all penalties in our given range. This chapter is published as the journal article *Computationally Efficient Changepoint Detection for a Range of Penalties* in Journal of Computational and Graphical Statistics (Haynes et al., 2017).

The penalised optimisation approach requires a specified cost for the segments. The optimal number and location of the changepoints are then found by minimising the total segmentation cost over a different number and location of changepoints. In Chapter 4 we shift our focus to this cost function. In particular many cost functions in the literature require assumptions of the underlying distribution of the data. We propose a new algorithm, ED-PELT (PELT with an Empirical Distribution cost function), which uses a cost function based on the empirical distribution of the data. This approach is nonparametric and hence can be applied to a large variety of data-sets,

since we do not require any prior knowledge of the parameters. We apply this method to heart-rate data recorded during a period of physical activity and show that ED-PELT works better than using a cost function which assumes the data is normally distributed. This chapter is published as the journal article *A computationally efficient nonparametric approach for changepoint detection* in Statistics and Computing (Haynes et al., 2016).

There are two main approaches for solving the optimisation problem in changepoint detection. The first is an approximate approach which involves recursively detecting single changepoints. The second uses dynamic programming which is an exact approach but can be computationally expensive and therefore do not scale well to large amounts of data. In Chapter 5 we show how we can utilise High Performance computing by parallelising the changepoint detection algorithms. We look at various ways to split the data across multiple cores and then merge the results without losing much accuracy.

We conclude this thesis with a discussion of the main contributions and discuss potential areas for further research in Chapter 6.

Chapter 2

Review of the Changepoint Literature

In this Chapter we will review some of the changepoint literature, in particular we will look at methods for offline, multiple changepoint detection in univariate data. There are many researchers across a vast range of disciplines using and developing state of the art algorithms for changepoint detection. Changepoint detection was first used for quality control (Page, 1954) but ever since changepoints have been of interest across many different fields. Below are some examples of where the detection of changes can play an important, and sometimes even life changing, role. This is by no means an exhaustive list but it does highlight the variety of applications in which changepoint detection can be, and has been, used.

2.1 Applications

Genomics

In genomics changepoint detection has been used for DNA sequencing to identify pattens in the gene (Braun and Müeller, 1998; Braun et al., 2000). The DNA copy number of a region is the number of copies of genomic DNA. In humans, this copy

number is two for all of the autosomes. To find the copy numbers on the genome, techniques such as comparative genomic hybridization (CGH) or array-based comparative genomic hybridization (array-CGH), for higher resolutions, are used. Changepoint detection is then used to find the regions where there is loss or amplification in tumour cells (Picard et al., 2004; Zhang and Siegmund, 2007). Variations in the copy number are common in cancer and other diseases (Olshen et al., 2004). Hocking et al. (2013b) review different methods for detecting changes in the chromosomal copy number. More recently Cleynen et al. (2013) apply changepoint detection to RNA-seq data which they claim will have improved accuracy over the use of CGH arrays. Cleynen et al. (2014) provide a recent comparison of segmentation methods on RNA-Seq data.

Environmental

In an environmental setting, changepoint detection can prove to be useful for logistical reasons, such as maintenance scheduling or managing resources. Wang et al. (2014) use changepoint detection to see if there has been changes in the monthly precipitation at various watersheds, across Southeast United States, due to climate change. In oceanography, Killick et al. (2013) and Nam et al. (2015) detect changes, and determine the uncertainty, in the autocovariance of wave heights, in the North Sea, to determine storm season changes. Reeves et al. (2007) provide a comparison of techniques used to detect changes in climate data, specifically they show examples of detecting changepoints in annual average temperature recorded in Alabama and Montana.

Finance and Economics

Financial data-sets, such as emerging stock markets and asset returns, can be extremely volatile. Although models such as the GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model account for conditional changes in the variance, Lamoureux and Lastrapes (1990) show that the persistence of variance may be over-

stated by not accounting for the breakpoints in the volatility. Finding the changes in volatility is required for risk management, forecasting and hedging (Fernandez, 2004). For example Aggarwal et al. (1999) use the changepoint detection method of Inclán and Tiao (1994) to detect shifts in volatility in emerging markets, before examining the local and global events which happened at the time of the changes. Andreou and Ghysels (2009) discuss the implications of ignoring changepoints in financial data as well as review a number of different changepoint detection tests used to detect different changes in financial asset returns and volatility. Other authors consider multivariate series of daily stock indices (such as Lavielle and Teyssière, 2006) or look at ways to detect changes as they happen in financial streams (Pepelyshev and Polunchenko, 2015).

Network Security

Cyber-attacks cost the UK economy billions of pounds every year. Computer networks can be represented as a data stream (an unending sequence of data-points) in which deviations from the normal network behaviour could be a sign of an attack. Sequential changepoint detection methods, in particular anomaly detection, can be used as intrusion detection systems. Various methods have been proposed that include detection in univariate streams (see for example Kim et al., 2004; Tartakovsky et al., 2013), Bayesian methods (Heard et al., 2010), nonparametric methods (Lévy-Leduc and Roueff, 2009) and detection in multivariate streams (Bodenham and Adams, 2013). Changepoint models have also been used to detect the attack of instant messaging worms (Yan et al., 2008).

Other

The above examples crop up frequently in changepoint detection however there are a huge range of other applications. In neuroscience characterising relative blood flow changes from fMRI scans of the brain are of interest (Aston and Kirch, 2012) as is

the detection of changes in brain signals such as electroencephalograms (EEG) which can be used to understand the cognitive processes in response to external stimuli (Kirch et al., 2015). When drilling for oil it is useful to detect changes in rock type to prevent blow-outs. This can be achieved by measuring and detecting changes in nuclear magnetic response (Fearnhead and Clifford, 2003). Our final example is in linguistics, where changepoint detection can be used to track shifts in meaning and usage of words through time (Kulkarni et al., 2015).

Now that we have shown the breadth of applications where changepoint detection can be useful, we will review the literature on some of the methods to detect changes. Again we will only cover a subset of the literature, given how vast it is, but we refer you to the following review papers, books and book chapters for further methods: Chen and Gupta (2000), Eckley et al. (2011), Jandhyala et al. (2013).

The structure of the rest of this chapter is as follows. In the first instance we introduce the changepoint model, in particular we will show how the multiple changepoint problem can be extended from single changepoint. This includes an introduction to Binary Segmentation-type methods, in Section 2.4, which recursively detect single changes on subsets of the data. In Section 2.5 we will look at how changepoint detection can be viewed as an optimisation problem that can be solved by either constraining the number, or the maximum number, of changes to be detected or by adding a penalty for every detected change. This thesis has a strong emphasis in dynamic programming methods and in Section 2.5.2 we will give a brief introduction to these methods.

Many methods require assumptions based on the underlying signal distribution which in reality may be unknown. In order to extend the dynamic programming methods to a wider range of applications we need to develop cost functions that do not rely on the distributional parameters. In Section 2.6 we will review the nonparametric changepoint literature.

This thesis is concentrated around a narrow area of changepoint detection but the

literature is extremely vast. In Section 2.7 we will discuss alternative methods to give a bit of an insight of what else is out there. This section includes a review of Bayesian, multivariate and online changepoint detection.

One main goal of this PhD is to develop methods that can be used in Big Data and thus the computational complexity of dynamic programming needs to be addressed. With computational power increasing and high performance computer power being easier to access we are interested in developing ways to run algorithms in parallel. In Section 2.8 we will review the literature on high performance computing and parallel algorithms.

2.2 Model

This thesis focusses on multiple changepoint detection. The general model we will use, unless otherwise stated, is: assume we have some data-series $y_1, ..., y_n$ ordered based on some covariate information such as time or position along a chromosome. This data-series will have m changepoints at locations $\tau_{1:m}$ where $\boldsymbol{\tau} = \{0 = \tau_0 < \tau_1 < ... < \tau_m < \tau_{m+1} = n\}$. Thus the changepoints will split the data in $m + 1$ segments where the ith segment contains the data-points $y_{(\tau_{i-1}+1):\tau_i}$. That is we assume the data is left-continuous. Although not completely analogous, we will show how the multiple changepoint detection model stems from the single changepoint detection case.

2.3 Single Changepoint Detection

In single changepoint detection we want to choose the best model either with no changepoint, $(m = 0)$ or one changepoint at location τ $(m = 1)$, where m denotes the number of changepoints and τ splits the data into two distinct segments, $y_{1:\tau}$ and $y_{(\tau+1):n}$. This is essentially a model selection problem.

A natural approach to the model selection problem is to perform a hypothesis test where H_0 is no changepoint $(m = 0)$ and H_1 is there is a single changepoint

($m = 1$). To test for a changepoint we can use the likelihood-ratio approach which was first proposed for use in this scenario by Hinkley (1970) who applied this method to detect changes in the mean in normally distributed data. This approach has also been applied to detect changes in data generated from different distributions such as exponential (Haccou et al., 1987) and binomial (Hinkley and Hinkley, 1970) as well as used to detect changes in variance in normally distributed data (Chen and Gupta, 1997).

The likelihood-ratio approach requires the calculation of maximum log-likelihoods under both the null and alternative hypotheses. Under the null hypothesis the maximum log-likelihood is just $l(y_{1:n}|\hat{\theta})$, where $l(\cdot)$ is the log-likelihood of the probability density function and $\hat{\theta}$ is the maximum likelihood estimator for the parameters. The maximum log-likelihood under the alternative hypothesis is given by

$$\max_{1\leq\tau<n}\left\{l(y_{1:\tau}|\hat{\theta}_1)+l(y_{(\tau+1):n}|\hat{\theta}_2)\right\}, \tag{2.1}$$

where $\hat{\theta}_1$ and $\hat{\theta}_2$ are the maximum likelihood estimators for the data before and after the changepoint respectively. The log-likelihood test statistic is then

$$\lambda = 2\left[\max_{1\leq\tau<n}\left\{l(y_{1:\tau}|\hat{\theta}_1)+l(y_{(\tau+1):n}|\hat{\theta}_2)\right\}-l(y_{1:n}|\hat{\theta})\right], \tag{2.2}$$

where the null hypothesis is rejected if $\lambda > c$ for some threshold value c. If a changepoint is found then the position of the changepoint, τ is estimated by

$$\hat{\tau} = \arg\max_{1\leq\tau<n}\left\{l(y_{1:\tau}|\hat{\theta}_1)+l(y_{(\tau+1):n}|\hat{\theta}_2)\right\}. \tag{2.3}$$

In standard hypothesis testing the threshold, c, is chosen such that it bounds the Type I error rate, however in the changepoint setting the likelihood function is discontinuous and thus is not twice continuously differentiable. This violates the assumptions required to be able to use a chi-squared distribution for the asymptotic distribution

of λ. Approximate thresholds can be calculated using the asymptotic distibutions of the likelihood functions (Chen and Gupta, 2000).

2.4 Binary Segmentation and Variants

The log-likelihood ratio approach only detects a single changepoint and the multiple changepoint detection problem cannot be formulated in this way. However, there is a subset of multiple changepoint algorithms which recursively perform single changepoint detection. The best known algorithm in this category is Binary Segmentation (BS) which was introduced by Scott and Knott (1974) and first applied in a stochastic setting by Vostrikova (1981). In BS the whole data-set is searched over to detect the location of a single changepoint. If we rewrite (2.2), then this is the point, τ, that satisfies the condition in (2.4) and also maximises the left hand side of (2.4).

$$l(y_{1:\tau}|\hat{\theta}_1) + l(y_{\tau+1:n}|\hat{\theta}_2) - c > l(y_{1:n}|\hat{\theta}). \tag{2.4}$$

The data is split at τ and the process is repeated on the segments $y_{1:\tau}$ and $y_{\tau+1:n}$. This process continues until no further changes are found. BS is a computationally efficient algorithm with computational cost $\mathcal{O}(n \log n)$ however it struggles to detect short segments especially if the data then returns to the pre-change distribution after the segment. For example, Fryzlewicz (2014) look at the asymptotic properties of BS with the cumulative sums (CUSUM) test statistic (Page, 1954) and show that as the number of data-points, $n \to \infty$, then BS is only asymptotically guaranteed to identify the true changepoints if the minimum segment length is $\mathcal{O}(n^{3/4})$.

2.4.1 Wild Binary Segmentation

Fryzlewicz (2014) attempt to overcome the weakness in the consistency of BS by introducing the Wild Binary Segmentation (WBS) algorithm. At each stage of BS, instead of calculating the global cost $\mathcal{C}(y_{1:n})$, WBS randomly draws a number of sub-

samples, $y_{s:e}$, where $1 \leq s < e \leq n$, and detects a candidate changepoint within each sub-sample. The changepoint within each sub-sample that has the largest likelihood-ratio value is found to be the new changepoint, τ. The data is split at τ and the process is repeated, similar to BS. By localising the costs, WBS overcomes the issue of changes being undetected in BS when they are too close to other changes.

If the number of sub-samples is suitably large then Fryzlewicz (2014) claims that, with high enough probability, there will be a sub-sample that only contains one changepoint at a suitable distance away from the end points. This localised feature will make it easier for detecting changes that may well be missed when looking over the whole data-set. Given a suitably chosen number of sub-samples and the CUSUM test statistic, Fryzlewicz (2014) show that WBS produces consistent results even when the minimum segment length is $\mathcal{O}(\log(n))$. The additional computational complexity of WBS over BS will depend on the number of sub-samples chosen to be calculated at each stage. Karolos and Fryzlewicz (2016) extend this method, by combining the CUSUM test statistics obtained at different scales of the wavelet periodogram, to detect changes in the second order structure of a piecewise stationary time-series.

2.4.2 Circular Binary Segmentation

Another approach used to overcome the limitation of BS when there are two points close to each other is Circular Binary Segmentation (CBS) proposed by Olshen et al. (2004). CBS uses an alternative test statistic (Levin and Kline, 1985) which searches for two changepoints, unlike a single changepoint in the standard BS. This test statistic assumes the means before the first changepoint and after the last changepoint are the same, and thus can be considered in a circle. The test statistic then tests whether the mean of the arc between the changepoints is different to the compliment. This is a recursive process that continues until no further changes are found.

One issue with CBS is that if either of the best two changes are found to be too close to the edge of the segment then there may only be one changepoint in the

segment. In this case each of the changepoints' viability is checked.

To generalise CBS to non-normal data Olshen et al. (2004) uses a permutation approach to calculate the p-values from reference distributions. This approach is computationally expensive as the number of permutations required is $\mathcal{O}(n^2)$. For large data-sets they suggest a window approach which divides the data into overlapping windows of equal size and searches for changes in each window.

Venkatraman and Olshen (2007) propose two ways to speed up the computation of CBS. The first is a hybrid approach, that uses a tail probability approximation for the maxima of a Gaussian Random field, to calculate the p-values. The second is a way of reducing the number of permutations when there is strong evidence of a change.

2.5 Optimisation Problem

The log-likelihood approach to changepoint detection can be adapted to the multiple changepoint case via a penalised cost approach. If we reformulate slightly and define the cost of a segment to be twice the negative maximum log-likelihood, i.e. $\mathcal{C}(y_{s:t}) = -2\max_\theta l(y_{s:t}|\theta)$ for any $t > s$ then the likelihood-ratio test (2.4) can be expressed as

$$\mathcal{C}(y_{1:\tau}) + \mathcal{C}(y_{\tau+1:n}) + \beta < \mathcal{C}(y_{1:n}), \tag{2.5}$$

where we have redefined the threshold c as a penalty β. That is, for the single changepoint case we want to solve

$$\min_{1\leq\tau<n} \left\{\mathcal{C}(y_{1:n}), \mathcal{C}(y_{1:\tau}) + \mathcal{C}(y_{\tau+1:n}) + \beta\right\}. \tag{2.6}$$

In the multiple changepoint setting this can be extended to solve for the number and location of changepoints. For example to solve for a maximum of 2 changepoints we

have

$$\min_{1\leq\tau_1<\tau_2\leq n}\left\{\mathcal{C}(y_{1:n}),\mathcal{C}(y_{1:\tau_1})+\mathcal{C}(y_{\tau_1+1:n})+\beta,\mathcal{C}(y_{1:\tau_1})+\mathcal{C}(y_{\tau_1+1:\tau_2})+\mathcal{C}(y_{\tau_2+1:n})+2\beta\right\}. \tag{2.7}$$

If we also wish to infer the number, m of changepoints, then this suggests solving

$$Q(y_{1:n},\beta)=\min_{m,\tau_{1:m}}\left\{\sum_{i=1}^{m+1}[\mathcal{C}(y_{(\tau_{i-1}+1):\tau_i})+\beta]\right\}. \tag{2.8}$$

The number and location of the changepoints are jointly estimated by finding the minimum segmentation cost. This is referred to as a *penalised minimisation problem*, since for every changepoint detected a penalty is added to avoid over-fitting.

Alternatively if the number of changepoints to be detected is pre-determined we can directly solve a *constrained minimisation problem.* This is

$$Q_m(y_{1:n})=\min_{\tau_{1:m}}\left\{\sum_{i=1}^{m+1}[\mathcal{C}(y_{(\tau_{i-1}+1):\tau_i})]\right\}. \tag{2.9}$$

It is unlikely in practice that the number of changepoints will be known however we might have an idea of the maximum number of changes which as will define as M. In this case we can solve (2.9) for $1{:}M$ and then solve

$$\min_{m\in\{1:M\}}\left\{Q_m(y_{1:n})+\gamma(m)\right\}, \tag{2.10}$$

where $\gamma(m)$ is a suitably chosen penalty term that increases with m. If $\gamma(m)$ is a linear function, that is $\gamma(m)=(m+1)\beta$ for some $\beta>0$, then this is analogous to the penalised minimisation problem.

2.5.1 Cost Functions

In the above formulation we take the segmentation costs to be the maximum log-likelihoods. That is, if we have data in a segment $y_{(s+1):t}$ drawn from a Gaussian

distribution with a common variance, σ^2, and segment specific mean, μ, then the segment cost taken from twice the negative log-likelihood will be

$$\mathcal{C}(y_{(s+1):t}) = -2 \times \frac{-1}{2\sigma^2} \sum_{j=s+1}^{t} (y_j - \hat{\mu})^2 \approx \sum_{j=s+1}^{t} (y_j)^2 - \frac{\left(\sum_{i=s+1}^{t} y_i\right)^2}{(t-s)}, \tag{2.11}$$

where $\hat{\mu}$ is the maximum likelihood estimator for the segment mean. Here we ignore the multiplicative constants since, if we re-define the penalty accordingly, these will not affect the optimisation problem.

Similarly if we have a fixed mean μ and segment specific variance, σ^2 then the segment cost will be

$$\mathcal{C}(y_{(s+1):t}) \approx (t-s) \left[\log \left\{ \frac{1}{t-s} \sum_{j=s+1}^{t} (y_j - \mu)^2 \right\} + 1 \right]. \tag{2.12}$$

For completeness, if we have data with a segment specific mean, μ, and segment specific variance, σ^2, then the segment cost is

$$\mathcal{C}(y_{(s+1):t}) \approx (t-s) \left[\log \left\{ \frac{1}{t-s} \sum_{j=s+1}^{t} (y_j)^2 - \frac{(\sum_{i=s+1}^{t} y_i)^2}{t-s} \right\} + 1 \right]. \tag{2.13}$$

Here we have used twice the negative log-likelihood for the segment costs however the same method applies for other cost functions. Other common examples of this cost are cumulative sums (Page, 1954), quadratic loss (Rigaill, 2015; Inclán and Tiao, 1994) and minimum description length (Davis et al., 2006). Generally this cost requires modelling assumptions about the distribution of the data and the type of change we are attempting to detect. We will look at nonparametric approaches that do not require these assumptions in Section 2.6.

2.5.2 Dynamic Programming

The cost for the segments in the optimisation problems, in (2.9) and (2.8), are segment additive and thus the Bellman optimality principle holds (Bellman, 1957). This allows the use of dynamic programming methods to solve these optimisation problems.

Segment Neighbourhood Search

To solve the constrained problem in (2.9) Auger and Lawrence (1989) introduced the Segment Neighbourhood (SN) search method. This method involves specifying the maximum number of changes M and then finding the optimal segmentations with 1 to M changes. SN uses a recursion which links $Q_m(y_{1:t})$ to $Q_{m-1}(y_{1:s})$ for $s < t$. That is:

$$\begin{aligned} Q_m(y_{1:t}) &= \min_{\boldsymbol{\tau}} \left[\sum_{i=0}^{m} \mathcal{C}(y_{(\tau_i+1):\tau_{i+1}}) \right], \\ &= \min_{\tau_m} \left[\min_{\tau_{1:(m-1)}} \sum_{i=0}^{m-1} \mathcal{C}(y_{(\tau_i+1):\tau_{i+1}}) + \mathcal{C}(y_{(\tau_m+1):\tau_{m+1}}) \right], \\ &= \min_{\tau_{m-1}} \left[Q_m(y_{1:s}) + \mathcal{C}(y_{(\tau_m+1):\tau_{m+1}}) \right]. \end{aligned} \tag{2.14}$$

The optimal segmentations for each number of changepoints is then found by a backwards recursion through the data. For each $t \in 1, ..., n$ the minimisation in (2.14) is calculated for all $s = 1, ..., t-1$. This has computation time $\mathcal{O}(n^2)$. This is repeated for all $m \in 1 : M$ and therefore SN had an overall computational cost of $\mathcal{O}(Mn^2)$. The quadratic cost means that this method is infeasible for large data-sets with a large number of possible changepoints.

Optimal Partitioning

Jackson et al. (2005) proposed a similar recursive method to SN to solve the penalised method in (2.8). For $t = 1, 2, ..., n$ their method, Optimal Partitioning (OP)

recursively solves

$$F(t) = \min_{\tau \in \boldsymbol{\tau_t}} \left\{ \sum_{i=1}^{m+1} [\mathcal{C}(y_{(\tau_{i-1})+1:\tau_i}) + \beta] \right\} = \min_{s \in \{0,\ldots,t-1\}} \{F(s) + \mathcal{C}(y_{(s+1):t}) + \beta\}, \quad (2.15)$$

where $\boldsymbol{\tau_t}$ is the set of all possible number and position of changepoints for segmenting the data up to time t. These recursions are solved with computational cost $\mathcal{O}(n^2)$. Extracting the set of changepoints in the optimal segmentation is achieved by a simple recursion backwards through the data. OP is much faster than SN however only 1 segmentation is found whereas SN can find a range of segmentations with $1 : M$ changes.

Pruning Techniques

To overcome the computational overhead of running dynamic programming algorithms there have been some recent algorithms that use pruning methods to reduce the computations. The two different types of pruning are *inequality based pruning* and *functional pruning*. The aim of both types of pruning is to remove the points that can never be changepoints from the space over which the recursions in (2.14) and (2.15) are performed.

Inequality based pruning To reduce the cost of OP, Killick et al. (2012) proposed the pruning method Pruned Exact Linear Time (PELT). This involves checking a single inequality condition to decide whether a candidate location for the most recent changepoint can be pruned. This has been defined as *inequality based* pruning in Maidstone et al. (2017).

Killick et al. (2012) show that if there exists a constant K such that for all $s < t < T$,

$$\mathcal{C}(y_{(s+1):t}) + \mathcal{C}(y_{(t+1):T}) + K \leq \mathcal{C}(y_{(s+1):T}), \quad (2.16)$$

and for $t > s$, if

$$F(s) + \mathcal{C}(y_{(s+1):t}) + K \geq F(t), \tag{2.17}$$

then at a future time $T > t$, s can never be the optimal last changepoint prior to T. The inequality in (2.16) is checked at time t for all current potential changepoints s. For all s for which (2.16) holds we prune s from our set of potential most recent changepoints going forward. PELT is implemented in the `changepoint` R package (Killick and Eckley, 2014; Killick et al., 2014).

Maidstone et al. (2017) propose a similar method where they apply inequality based pruning to SN (Segment Neighbourhood search with Inequality Pruning, SNIP), however this method is not competitive when compared to other pruned SN methods, introduced below.

Functional pruning The idea of functional pruning is to define the segmentation costs as a function over the segment parameter θ. To be able to do this we need to be able to split the segmentation costs into component parts, $\gamma(y_i, \theta)$. For the constrained case in (2.9) we define the new cost function $Cost_m^\tau(y_{1:t}, \theta)$ as the minimal cost of segmenting the data $y_{1:t}$ into m segments with the most recent changepoint at τ and the segment parameter after τ is θ. That is

$$Cost_m^\tau(y_{1:t}, \theta) = Q_{m-1}(y_{1:\tau}) + \sum_{i=\tau+1}^{t} \gamma(y_i, \theta), \tag{2.18}$$

This is the basis of the pruned dynamic programming algorithm (pDPA) proposed by Rigaill (2015) who develops a dynamic programming algorithm to update these recursively at each new time step.

Similarly Maidstone et al. (2017) apply functional pruning to the penalised optimisation problem (2.8) in their algorithm: Function Pruning Optimal Partitioning (FPOP). That is

$$Cost^\tau(y_{1:t}, \theta) = Q(y_{1:\tau}, \beta) + \sum_{i=\tau+1}^{t} \gamma(y_i, \theta), \tag{2.19}$$

where $Q(y_{1:\tau}, \beta)$ is the optimal segmentation prior to τ, i.e.

$$Q(y_{1:\tau}, \beta) = \min_{\tau_1} Cost^{\tau_1}(y_{1:\tau}, \theta). \tag{2.20}$$

For both algorithms these functions only need to be stored for the candidate changepoints and are recursively updated at each time point,

$$Cost^{\tau}(y_{1:t}, \theta) = Cost^{\tau}(y_{1:(t-1)}, \theta) + \gamma(y_t, \theta). \tag{2.21}$$

The minimum cost of segmenting data $y_{1:t}$, conditional on the last segment having parameter θ is

$$Cost^{*}(y_{1:t}, \theta) = \min_{\theta} Cost^{\tau}(y_{1:t}, \theta). \tag{2.22}$$

The functions are point additive and thus it is theoretically possible to prune sets of segmentations. If a potential last changepoint, τ_1, does not form part of the piecewise function $Cost^{*}(y_{1:t}, \theta)$ for a time t, i.e. there does not exist a θ such that $Cost^{*}(y_{1:t}, \theta) = Cost^{\tau_1}(y_{1:t}, \theta)$, then at future time points this will still be the case and thus τ_1 can never be the most recent change. The function $Cost^{\tau_1}(y_{1:t}, \theta)$ can be pruned as it will never be optimal, hence the term functional pruning.

For the quadratic loss function with normally distributed data, Rigaill (2015) show that this functional pruning is efficient and they empirically show that this method has sub-quadratic time in $\mathcal{O}(n \log n)$. Further implementation of pDPA applied to RNA-Seq data with a negative Binomial model has been looked at by Cleynen et al. (2013). Although not implicitly shown, Cleynen et al. (2013) also say their results hold for the Poisson model.

PDPA needs to store the $Cost_m^{\tau}(y_{1:t}, \mu)$ functions as well as the candidate changepoint set for all $m = 1, ..., M$ and therefore the computational complexity is similar to SN. Since FPOP uses OP it is computationally faster. PDPA is implemented in the `cghseg` package (Picard et al., 2016) for the quadratic loss function and in

`Segmentor3IsBack` (Cleynen et al., 2013) which includes the negative Binomial and the Poisson model.

At the time this research project commenced PELT was arguably the best method for changepoint detection using dynamic programming due to its speed advantages over SN and pDPA. Many of the methods in this thesis have been developed around PELT so I will look further into this method later in this thesis. FPOP was developed by colleagues at Lancaster during this PhD and has been shown to outperform PELT in the case of a change in mean. Maidstone et al. (2017) show that functional pruning always prunes more than inequality based pruning and this is especially the case where there are few changes. FPOP does not work when the segment parameter θ has dimension greater than one and thus PELT is still the superior method to use in cases where we have changes in more than one parameter such as mean and variance.

2.5.3 Penalties

In the algorithms which use the optimisation problem the choice of the penalty parameter, β, has a significant impact on the accuracy of the detected changes. If we let p denote the number of additional parameters introduced by adding a changepoint, then popular examples used frequently in the literature include $\beta = 2p$ (Akaike's Information Criterion (AIC); Akaike, 1974), $\beta = p \log n$ (Schwarz's Information Criterion (SIC/BIC); Schwarz, 1978) and $\beta = 2p \log\log n$ (Hannan-Quinn; Hannan and Quinn, 1979). The AIC is the simplest penalty choice but it usually leads to over-fitting the data. The Hannan-Quinn penalty also normally leads to over-fitting, this is due to these penalties being small, even for large n. Yao (1988) establish weak consistencies for estimating the number and position of changepoints, in normally distributed data, using the SIC penalty.

More sophisticated penalty terms have been proposed. Zhang and Siegmund (2007) propose a method which accounts for the length of the segments (Modified Bayesian Information Criterion, mBIC). The mBIC is shown to work well for simu-

lated data where the model assumptions of the mBIC hold however it does not work as well for real data-sets (Hocking et al., 2013b). Lavielle (2005) propose an adaptive penalty choice. This involves solving the constrained optimisation problem for different number of changepoints. They then plot the unpenalised cost against the number of changepoints detected and suggest the point that lies on the "elbow" of this plot to be the one with the best segmentations. Intuitively this is the point where the cost stops decreasing as much with an addition of a false changepoint. In a similar fashion Hocking et al. (2013a) calculate the optimal segmentations with different numbers of changepoints and then use annotated training data to learn the best choice of penalty.

It is common that the penalty is linear in the number of changepoints however there are important exceptions, for example the Minimum Description Length. Davis et al. (2006) and Li and Lund (2012) use the Minimum Description Length (MDL) penalty, proposed by Rissanen (1989) to detect changepoints. This penalty arises from information theory and essentially finds the model which gives the best compression of the data. That is to store the data, the data is split up and the best model is the one that requires the least amount of space (i.e. smallest code length) to store the data. For a model with parameters $\theta_{1:m}$ the MDL is

$$MDL(m, \tau_{1:m}, \theta_{1:m}) = \log m + (m+1)\log n + \sum_{j=1}^{m+1}\left[\log\theta_j + \frac{\theta_j + 2}{2}\log(\tau_j - \tau_{j-1}) + \frac{(\tau_j - \tau_{j-1})}{2}\log(2\pi\hat{\sigma}_j^2)\right], \tag{2.23}$$

where $\hat{\sigma}^2$ is the Yule-Walker estimate of σ^2. This is essentially the code length for the model plus the code length for the residuals, used to assess the fit of the model. Rissanen (1989) shows that the code length for the residuals is equal to the negative log-likelihood of that model. A more complex model implies a larger encoding cost and therefore a larger penalty. As such the penalty term for the MDL is equal to the cost of encoding the model.

Killick et al. (2012) show that solving the linear penalty case with the correct penalty will give the optimal solution for many non-linear penalties.

2.5.4 Simultaneous Multiscale Changepoint Estimator

Frick et al. (2014) use dynamic programming to minimise a multiscale statistic for a range of step functions in their method SMUCE (Simultaneous Multiscale Changepoint Estimator). SMUCE is used to detect changepoints in exponential regression and works by minimising the number of changepoints over the acceptance region of a multiscale test at a level α. As well as the number and location of changepoints, SMUCE is able to estimate confidence bands for the step function representing the underlying signal as well as confidence bands for the estimated changepoint locations.

The main disadvantage of SMUCE is that it only allows for detection in a single parameter. There has been various adaptions of SMUCE: Pein et al. (2015) extend SMUCE to work on heterogeneous data, where at a changepoint the variance also changes (H-SMUCE), Futschik et al. (2014) apply SMUCE to DNA segmentation which follows a Bernoulli distribution (B-SMUCE) and Hotz et al. (2013) extends SMUCE for dependent Gaussian data.

2.6 Nonparametric Approaches

For many cost functions, $\mathcal{C}(\cdot)$, knowledge of the underlying distribution of the data is required. For example, in the likelihood methods we need to know the distribution of the data in order to formulate the likelihood and to find maximum likelihood estimators. In practice, however, we might not know the underlying distribution or the data might not even follow a standard distribution. Mis-specifying models can have detrimental effects on the performance of the changepoint detection methods and thus there is interest in developing methods that do not have any assumptions on the distribution of the data.

There has been a vast amount of work on single changepoint detection in the non-parametric setting. The first ever changepoint test, CUSUM (cumulative sums), proposed by Page (1954) is a nonparametric approach. Further work in single changepoint detection includes Bhattacharyya and Johnson (1968); Carlstein (1988) and Dümbgen (1991). Discussion on nonparametric methods and some general asymptotic results can be found in Brodsky and Darkhovsky (1993) and Csörgö and Horváth (1997).

Many of the nonparametric test statistics use ranks of the observations where the rank of the ith observation at time t can be defined as

$$r(x_i) = \sum_{i \neq j}^{t} \mathbb{1}(x_i \geq x_j), \tag{2.24}$$

where $\mathbb{1}$ is an indicator function. For example Pettitt (1979) and Hawkins and Deng (2010) use a Mann-Whitney test statistic to detect changes in location. The test statistic is calculated as

$$\lambda_\tau = 2 \sum_{i}^{\tau} r(x_i) - \tau(n+1), \tag{2.25}$$

and is computed for all values $1 < \tau < n$. A changepoint is detected if the maximum exceeds some threshold where the maximum is thus the detected changepoint.

Similarly for a change in scale the Mood test statistic can be used (Mood, 1954). The test statistic in this case is

$$\lambda_\tau = \sum_{i}^{\tau} (r(x_i) - (n+1)/2)^2, \tag{2.26}$$

and again is computed for all values $1 < \tau < n$.

For a more general test for changes in location and scale, Ross and Adams (2012) discuss the use of the Kolmogorov-Smirnov and the Cramer-von Mises statistics. Both of these compare the empirical distribution of the data before and after a change. If we define S_1 as the sample before the change and S_2 as the sample after, the empirical

distributions are calculated as

$$\hat{F}_{S_1}(x) = \frac{1}{\tau}\sum_{i=1}^{\tau} \mathbb{1}(X_i \leq x) \tag{2.27}$$

$$\hat{F}_{S_2}(x) = \frac{1}{n-\tau}\sum_{i=\tau+1}^{n} \mathbb{1}(X_i \leq x). \tag{2.28}$$

For the Kolmogorov-Smirnov test, the test statistic is defined as

$$\lambda = \sup_x |\hat{F}_{S_1}(x) - \hat{F}_{S_2}(x)|, \tag{2.29}$$

and for the Cramer-von-Mises test statistic this can be calculated by

$$\lambda = \sum_{i=1}^{n} |\hat{F}_{S_1}(x) - \hat{F}_{S_2}(x)|^2. \tag{2.30}$$

The use of the empirical distribution has also been used in other methods. Guan (2004) use an empirical likelihood ratio test to propose a semi-parametric approach to detect a change from a distribution to a weighted one. Without assuming any relationship between the two populations, Zou et al. (2007) use the empirical likelihood to develop a fully nonparametric approach. They show that the asymptotic properties are similar to those of the parametric likelihood methods. Other methods include using Kernel density estimations (Baron, 2000), however these methods are computationally intensive.

Extending these methods to detect multiple changepoints is not straightforward. Within sequential changepoint detection this can be treated as a single changepoint problem that resets every time a changepoint is detected (Ross and Adams, 2012). Lee (1996) proposed a weighted empirical measure which essentially uses single changepoint detection over a window of observations and then runs the window through the entire data. This method is simple to use but it lacks in performance in terms of the number and location of changepoints detected. Zou et al. (2014) then developed a

method using the empirical distribution as a cost function with Segment Neighbourhood search and show, under mild conditions, that the consistency of the detected changepoints is $\mathcal{O}_p(1)$. The issue with this approach is the high computational cost which is $\mathcal{O}(mn^2 + n^3)$, where m is the number of changepoints. We will explore this method further in Chapter 4.

2.7 Other Approaches

In this thesis we focus on detecting changes in univariate time-series. We have adopted frequentist approaches to detect changes in the offline setting, that is we already have access to the full data-set. There are many important areas in the changepoint literature which are worth noting. In particular these include Bayesian methods, multivariate changepoint detection and online/sequential detection. Below we will briefly introduce these methods and highlight notable works in each area.

2.7.1 Bayesian Methods

Bayesian techniques for changepoint detection require priors for the number and location of changepoints, as well as for the segment parameters. Bayesian techniques based on Markov chain Monte Carlo, MCMC, have been used for inference of changepoint models (Stephens, 1994; Chib, 1996, 1998). When the number of changes is unknown a common approach is reversible jump MCMC proposed in Green (1995) which explores the joint space of the model and parameters for a set of models with different number of changepoints. Lavielle and Lebarbier (2001) propose a hybrid method using the Metropolis-Hastings algorithm with a Gibbs-sampler and show that this converges much faster than the reversible jump algorithm in Green (1995). The difficulty with the MCMC methods is finding moves that allow the algorithm to mix well as well as being able to determine if the algorithm has converged.

Alternatively, there are methods which directly simulate from the posterior based

on an exact method for calculating the posterior means (Barry and Hartigan, 1992). This method was used by Liu and Lawrence (1999) for DNA sequencing and has since been used more generally in Fearnhead (2005) and Fearnhead (2006). Fearnhead and Liu (2007) apply this method for online changepoint detection and shows this to have a cost linear in the number of observations. However these methods require the parameters within a segment to be independent of each other and that the marginal likelihood for the data within each segment can be calculated. Fearnhead and Liu (2011) extend the direct simulation approach to models where there is dependence across segments. They develop an online Bayesian approach which can be used under the assumption that the dependence of the parameters is Markov (the parameters of the current segment depend only on the previous segments).

An alternative Bayesian approach for online changepoint detection was proposed by Adams and MacKay (2007) who use the posterior distribution for the number of data observed since the last changepoint, i.e., the current "run length", to predict the distribution of the next data-point conditional on the run length. They apply this method to detect changes in rock strata, Dow Jones returns and coal mine explosions.

2.7.2 Hidden Markov Models

Analogous to the Bayesian methods, Hidden Markov Models, HMMs, (see Cappé et al., 2005, for an overview) can also be used for changepoint detection. For changepoint detection the data are the observations and the hidden underlying states are the segmentations. Luong et al. (2012) provide an introduction to using HMMs for changepoint detection. HMMs have been used within classical forward-backward recursions (Durbin et al., 1998) to calculate the posterior marginal state distribution as well as in the expectation-maximisation algorithm (Dempster et al., 1977) for estimation in mixture and changepoint problems. HMMs have also been used within MCMC algorithms such as in reversible jump MCMC (Green, 1995).

There are many methods which have been proposed for changepoint detection

within the HMM framework. For example Nam et al. (2012) use sequential MCMC to detect changes in fMRI data and Nason et al. (2000) detect changes in autocovariance using Locally Stationary Wavelets.

There has also been work on estimating the number of hidden states in the HMM. Zhang and Siegmund (2007) use their modified Bayes Information Criterion to adjust for the number of states in the previous HMM and Picard et al. (2004) use an adaptive method to estimate the number and location of changepoints.

2.7.3 Multivariate Methods

In some applications there may be multivariate time-series where changes occur either simultaneously in all of the variables, *fully mutivariate*, or in a subset of the variables, *subset multivariate*. For example in financial markets it has been shown that several time-series have the same changes in volatility (Teyssière, 2003) whereas in DNA copy number variation often the DNA variations only occur in the proportion of the samples (see for example Bardwell and Fearnhead, 2017).

Each of the series could be analysed using univariate methods, however ignoring the other series will result in a loss of power. Thus multivariate methods which take into account all of the variables simultaneously are of interest.

Fully Multivariate

Traditionally, methods for changepoint detection in multivariate data are fully multivariate since this case is often simpler than detecting changes in subsets of variables. One of the earliest approaches for multivariate changepoint detection was by Srivastava and Worsley (1986) who detected a change in the mean vector of a multivariate normally distributed time-series. Single changepoint detection methods in the multivariate setting have also been proposed by Horváth and Hušková (2012) and Batsidis et al. (2013) who use parametric methods, and Aue et al. (2009) who propose a nonparametric approach.

Binary Segmentation can be adapted to use multiple dimensions (Srivastava and Worsley, 1986; Aue et al., 2009). Rather than extending the univariate counterpart, Matteson and James (2014) use Binary Segmentation at the core of their method: E-divisive. E-divisive is a nonparametric method based on hierarchical clustering and combines Binary Segmentation with a cost function based on the Euclidean distance between the observations over the multiple variables.

Alternatively, dynamic programming methods have been proposed. As in the univariate setting these methods require a cost for the multivariate time-series plus some penalty to avoid over-fitting. Lavielle and Teyssière (2006) and Maboudou and Hawkins (2009) propose multivariate methods based on Segment Neighbourhood Search using a penalised cost function. The calculations are similar to those in the univariate case but have additional $\mathcal{O}(p)$ calculations where p is the number of variables, hence it has an overall computational cost of $\mathcal{O}(Mpn^2)$. James and Matteson (2015) use the approach of Lavielle and Teyssière (2006) but with an approximation of the nonparametric test statistic used in the E-divisive method (Matteson and James, 2014). This approximation is used as a way to speed up the calculations. Another nonparametric approach was proposed by Lung-Yut-Fong et al. (2012) who use a nonparametric rank statistic as the cost in Segment Neighbourhood Search.

Other methods for fully multivariate changepoint detection have been proposed by Ombao et al. (2001) who use the SLEX (Smooth Localised Complex Exponentials) collection of bases to detect changes in the auto and cross correlation and Vert and Bleakley (2010) who use a LASSO based approach which fits a model to the total variation.

Subset Multivariate

The above methods make the assumption that all of the detected changes occur across all of the variables. However this is often not the case in practice. Cho and Fryzlewicz (2015) and Xie and Siegmund (2013) propose methods to detect a single change which

only affects a subset of the variables. Extending this to multiple changepoints, Zhang et al. (2010) and Siegmund et al. (2011) propose methods to detect changes in a large number, and a small number of variables, respectively. Jeng et al. (2013) develop a similar method to deal with both a large and small number of variables.

Using dynamic programming Maboudou-Tchao and Hawkins (2013) obtain a fully multivariate solution and perform a hypothesis test on each estimated changepoints to determine which variables it affects. Pickering (2015) proposes a method which minimises a cost function using an equivalent method to Optimal Partitioning. This method detects the changes and also finds the subsets affect at the same time which saves having the second step as in Maboudou-Tchao and Hawkins (2013). In the Bayesian framework Bardwell and Fearnhead (2017) propose a method using hidden states to detect changes in subsets of variables in copy number variation.

2.7.4 Online/Sequential Changepoint detection

The methods discussed so far have detected changepoints in scenarios where we have already recorded the entire data-set, this is known as *offline* detection. Modern technologies for recording data provide the opportunity to analyse data streams; data characterised by a potentially, unending sequence of high-frequency observations and thus there is vast literature on methods to detect changepoints sequentially ("online"). Online changepoint detection emerged from quality control where manufacturing processes were continuously monitored to detect an increase in the number of defective items (Page, 1954). Since then, sequential changepoint detection has been used in diverse applications such as fraud detection (Hand and Weston, 2008), finance (Wu et al., 2004) and computer networks (Bodenham and Adams, 2013).

Statistical Process Control

Traditionally online changepoint detection was referred to as statistical process control and it concerned data streams with only a single changepoint. The task in statistical

process control is to detect the change as soon as possible after they have occurred. The performance is usually measured using two criteria of the Average Run Length (ARL): the expected time between false positive detections (ARL_0) and the mean delay until a change is detected (ARL_1) (Page, 1954). When the pre-change distribution is known control charts such as the CUSUM algorithm (Page, 1954) and Exponentially Weighted Moving Average charts (Roberts, 2000) can be used. For an overview of these techniques see Basseville and Nikiforov (1993). Typically there is no prior knowledge of the distribution of the data stream hence nonparametric control charts have been developed. Several distribution charts have been proposed to monitor the location parameter, such as charts that use the Mann-Whitney/Wilcoxon Rank statistics (Chakraborti and van de Wiel, 2008; Hawkins and Deng, 2010).

Instead of comparing the observations to a known target value Hawkins et al. (2003) propose a changepoint control chart in which they treat the reference samples as part of the ongoing data stream. Hawkins et al. (2003) use this changepoint model framework to detect changes in mean in Gaussian data which has since been extended to changes in variance in Gaussian data (Hawkins and Zamba, 2005) and changes in mean of Bernoulli data (Ross et al., 2013). This has also been extended in the nonparametric framework to detect changes in location (Hawkins and Deng, 2010) and to detect changes in location and/or scale (Ross et al., 2011). Lai (2001) list a variety of changepoint models used for sequential detection in scenarios where some of the in-control parameters are known.

Continuous Monitoring

Since the changepoint control chart framework does not assume anything about the distribution before the data stream begins it can easily be extended to multiple changepoint detection by restarting the process once a change has been detected, and thus reducing the problem to successive detection of single changepoints. Intuitively this makes sense in scenarios where human intervention is required due to the change, such

as in process control the fault in the production line will need to be resolved and then the process can restart as if the change never occurred. In many real life applications the process normally continues even when a changepoint has occurred. For instance, in financial data streams detecting a change may trigger a trading action but the data stream will continue. The detection of multiple changepoints in this scenario is referred to as *continuous monitoring*.

One method for continuous monitoring is to assume that at the start of a regime the process is in control for a certain number of observations and to use these observations to estimate the parameters for the current regime (see for example Jones, 2002). This parameter estimation stage is referred to as the *burn-in* period. Adaptions of CUSUM and EWMA have also been considered for continuous monitoring (Apley and Chang-Ho, 2007; Jiang et al., 2008; Tsung and Wang, 2010; Capizzi and Masarotto, 2012), however these require numerous parameters to be calculated for practical application. Bodenham and Adams (2016) propose a method using adaptive forgetting factors to detect changes in location of data streams which only requires a single parameter to be selected.

2.8 High Performance Computing and Parallel Algorithms

We are living in an era where the amount of data we are collecting and storing is extremely large due to the revolution in technology. Data-sets with millions, or even billions, of observations are now common place but even those with tens of thousands present computational challenges. The increased length of data alongside the increased computational requirements, resulting from more complex algorithms and analysis, means there is a strong need for high performance computing. For the context of this thesis we are interested in the problem of dealing with longer data-sets, in particular those recorded using high frequency data sensors. Parallel computing

can help reduce the computational burden by using multiple processors or computers to share the work load.

2.8.1 Architecture

Modern computers have a parallel architecture with multiple processors/cores. To make the best use of the potential computer power we need to have a brief understanding of the hardware and the different communication tools required for the different architectures. Here we will give a high-level outline of some of the different hardware and software set-ups, for a more indepth introduction to the area see Tsuchiyama et al. (2010) and Barney (2016a).

Hardware

Parallelisation occurs at different levels of the hardware set-up. Multi-core and multi-processor computers have multiple processors on a single chip or machine. These architectures have a shared memory (Figure 2.1a) which allows all processors to communicate by reading/writing to a single memory. This is a very simple architecture from a software point of view, however it lacks scalability since if you add more processors this puts strain on the resources due to more processors trying to read/write to the same memory.

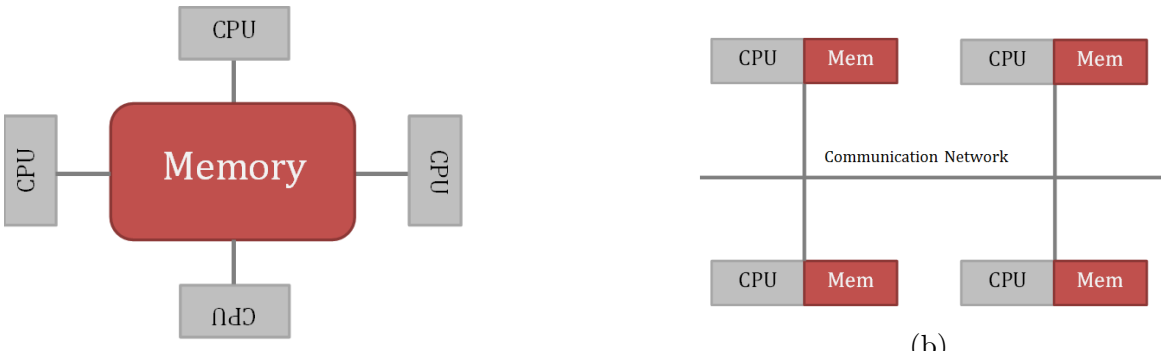


Figure 2.1: High-level examples of (a) shared and (b) distributed memory architectures.

Multi-computer systems such as computer clusters and grid/cloud computing are environments where multiple computers are connected together via a network. Each computer has its own memory and communicates through the network. This type of memory is known as distributed memory (Figure 2.1b) and is more favourable over shared memory as there are no bottlenecks associated with reading/writing to memory. The main difference in these systems is how they are connected in a network. In computer clusters the computer systems are connected locally using hardware, whereas computers in a grid or cloud network are connected via the internet. Generally clusters are made up of computers with similar hardware and operating systems whereas computers in a grid/cloud may have very different set-ups which need to be accounted for when developing software.

Using the maximum number of processors available will not necessarily be the best. Amdahl's law (Amdahl, 1967) states that the potential program speed up is

$$\text{speedup} = \frac{1}{1-p},$$

where p is the fraction of the code that can be parallelised. If we have L processors then this can be modelled by

$$\text{speedup} = \frac{1}{\frac{p}{L}+s},$$

where p is the fraction of the code that can be run in parallel and s is the serial fraction of the code. Figure 2.2 shows an illustration of Amdahl's law. There is also the additional communication cost to account for since the communication between processors is actually slower than computation.

Software

Various programming languages and libraries have been developed for the different parallel architectures. For shared memory computers OpenMP (Dagum and Menon,

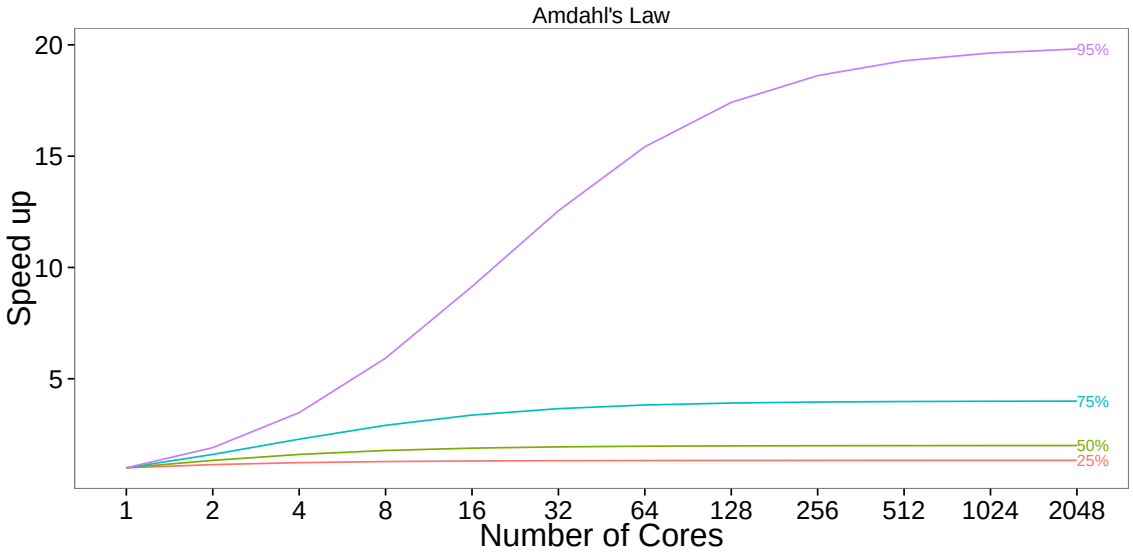


Figure 2.2: An illustration of Amdahl's law for different proportions of parallel code

1998) and POSIX threads (Butenhof, 1997, Pthreads) are the most common. Both of these methods use multi-threading where a master thread divides the tasks to a specified number of worker threads. In OpenMP the programmer highlights the section of code that is to be run in parallel and then the threads are formed before the section is executed. Alternatively, Pthreads allows the user to create, manipulate and manage threads and thus allows for more low-level control over the threads. OpenMP can be used with C/C++ and Fortran whereas Pthreads uses only C.

In distributed memory computers, message passing APIs (Application Programming Interfaces) are widely used. Two examples of which are MPI (Barney, 2016b, Message-Passing Interface) and PVM (Geist et al., 1994, Parallel Virtual Machine). In these systems one machine or processor becomes the "master" and controls the other devices "slaves". Jobs are distributed from the master process to the slaves, the slaves do their work and then send the results back to the master who combines the results from all of the slaves. MPI is the standard method however PVM allows for networks where the set-up across the machines is different.

2.8.2 R Packages

The computational aspects of this thesis will mainly be coded in R with some of the code having a C back-end. In Chapter 5 we develop methods for parallel changepoint detection which we will use R for the parallelisation. There are many packages that have been developed to provide communication to the various parallel infrastructures in R. For a review of some of these packages see Schmidberger et al. (2009) and for an up to date list of all the parallel packages in R see Eddelbuettel (2016). In this thesis we will use the `doParallel` package (Calaway et al., 2014) which provides a parallel back-end for the `foreach` package (Calaway et al., 2015) using the `parallel` package which is inside R-core. The `foreach` package allows general iterations over elements without using a loop counter and thus allows the loop to run in parallel. The `parallel` package started as a merger of the `multicore` and `snow` packages however most of the functionality of `multicore` has been integrated into `parallel`. The `snow` (Simple Network of Workstations) package in R (Tierney et al., 2015) supports several different low-level communications mechanisms including MPI, alongside PVM, NetWorkSpaces and raw sockets. This allows for the same code to run on clusters or on a single multi-core computer. The code we develop will be able to run on any parallel architecture, it will just require the user to modify the communication parts of the code to be specific to their parallel set up.

2.8.3 Parallel Algorithms

A common approach in high performance computing is to split the data into "chunks", run the analysis on each chunk in parallel and then somehow combine the chunks. The Binary Segmentation-type methods for changepoint detection discussed in Section 2.4 can easily be parallelised. That is at each step of algorithm the subsets of data can be analysed on a separate processor. This is what is known as "embarrassingly parallel"; that is it is embarrassing how easy it is to parallelise these methods. It is not as simple to parallelise the dynamic programming methods discussed in Section 2.5.2 since each

step of the algorithms are dependent of the previous steps. These are the methods, however, that would really benefit from being parallelised since the costs are at least $\mathcal{O}(n^2)$, if we ignore the situations where we can use pruning for the moment, and thus are computationally infeasible with large n.

Apart from the embarrassingly parallel Binary Segmentation type approaches to changepoint detection there is, to our knowledge, only one paper that address parallel changepoint detection in the univariate case. (There is research on multivariate changepoint detection that explores parallelisation but this deals with parallelisation over dimensions not observations). Nikol'skii and Furmanov (2016) propose a method which splits the data into equal sized segments and then simultaneously checks for a single changepoint on each subset of data. If no changepoints are found in adjacent segments then they take points around the segments to check for a changepoint. This is an approximate method which doesn't allow for multiple changepoints in the same subset of data. In fact this huge flaw makes this method statistically unsound as there is no guarantee of detecting all of the changepoints.

Across other areas of statistics there has been research in developing statistically sound methods for splitting and combining data across multiple processors. Matloff (2016) develop a broadly applicable "chunking and averaging" method for converting many non-embarrassingly parallel algorithms into embarrassingly parallel methods. This methods involves splitting the data into chunks, applying some sort of algorithm to each chunk, such as quantile regression, and then merging the chunks by averaging. Under the assumption that the data are IID, they show that asymptotically this method gives the same errors as the full estimator. This chunking approach was proposed by Hegland et al. (1999) for nonparametric regression modelling and has also been used by Fan et al. (2007) to overcome memory issues in linear regression for massive data-sets. In the case of Fan et al. (2007) they merge the data using a weighted average.

The above methods combine the data by averaging however this is not always

possible. Song and Liang (2015) use a "split and merge" approach for Bayesian variable selection for ultra high dimensional linear regression. In this case they split the data and perform Bayesian variable selection for each subset and then aggregate the variables that are selected from each subset. They then perform Bayesian variable selection on the aggregated subset. Similar split and merge approaches, occasionally referred to as "divide and conquer" have been used for other methods such as matrix factorization (Mackey et al., 2013), estimating equation estimation (Lin and Xi, 2011) and logistic regression (Xi et al., 2009).

Chapter 3

Computationally Efficient Changepoint Detection for a Range of Penalties

3.1 Introduction

Changepoints are considered to be those points in a data-sequence where we observe a change in the statistical properties. Assume we have data, $y_1, \ldots, y_n$, that have been ordered based on some covariate information, for example by time or by position along a chromosome. For clarity we will assume we have time-series data in the following. Our time-series will have m changepoints with locations $\tau_{1:m} = (\tau_1, ..., \tau_m)$ where each τ_i is an integer between 1 and $n-1$ inclusive. We assume that τ_i is the time of the ith changepoint, so that $\tau_1 < \tau_2 < ... < \tau_m$. We set $\tau_0 = 0$ and $\tau_{m+1} = n$ so that the changepoints split the data into $m+1$ segments with the ith segment containing the data-points $y_{(\tau_{i-1}+1):\tau_i} = (y_{\tau_{i-1}+1}, \ldots, y_{\tau_i})$.

There are many different approaches to changepoint detection; see Frick et al. (2014), Jandhyala et al. (2013), Fryzlewicz (2014) and references therein. One common approach is to define a cost for a given segmentation of the data such as the

negative log-likelihood (Chen and Gupta, 2000), quadratic loss (Rigaill, 2015) or the minimum descriptive length (Davis et al., 2006). Typically this cost is based on first defining a segment-specific cost function, which we denote as $\mathcal{C}(y_{(s+1):t})$ for a segment which contains data-points $y_{(s+1):t}$. We then sum this segment-specific cost function over the $m+1$ segments. A natural way to then estimate the number and position of the changepoints would be to minimise the resulting cost over all segmentations. Note that, whilst formulated differently, Binary Segmentation procedures (Scott and Knott, 1974; Olshen et al., 2004) can be viewed as approximately minimising such a cost (see Killick et al., 2012, for more discussion). From herein we use optimal in the sense that the segmentations are solutions of the constrained minimisation problem, i.e., if we have m changepoints then the location of these changepoints are such that they minimise the cost of segmenting the data with m changepoints.

Directly minimising such a cost function will generally result in over-fitting, as for many choices of cost function adding a changepoint always reduces the overall cost. There are two potential approaches to avoiding such over-fitting. The first of these would be to constrain the optimisation by fixing the maximum number of changepoints that can be found. The corresponding *constrained minimisation problem* is:

$$Q_m(y_{1:n}) = \min_{\tau_{1:m}} \left\{ \sum_{i=1}^{m+1} [\mathcal{C}(y_{(\tau_{i-1}+1):\tau_i})] \right\}, \tag{3.1}$$

with the best segmentation with m changepoints being the one that attains the minimum. If the number of changepoints is unknown then the number of changes, m, is often estimated by solving

$$\min_m \{Q_m(y_{1:n}) + f(m)\}, \tag{3.2}$$

where $f(m)$ is a suitably chosen penalty term that increases with m.

If $f(m)$ is a linear function, that is $f(m) = (m+1)\beta$ with $\beta > 0$, then we can jointly estimate the number and the position of the changepoints by solving a *penalised*

minimisation problem (see for example: Lavielle and Moulines, 2000; Lebarbier, 2005; Jackson et al., 2005; Boysen et al., 2009):

$$Q(y_{1:n}, \beta) = \min_{m, \tau_{1:m}} \left\{ \sum_{i=1}^{m+1} [\mathcal{C}(y_{(\tau_{i-1}+1):\tau_i}) + \beta] \right\}, \tag{3.3}$$

again with the estimated segmentation being the one that attains the minima. This second approach, of directly minimising (3.3) is computationally faster than solving the constrained penalisation problem for a range of the number of changepoints, and then minimising (3.2); however it requires a choice of penalty constant, β. Note that some choices of penalty include terms that depend on the segment lengths (e.g. Zhang and Siegmund, 2007; Davis et al., 2006). The resulting penalised minimisation problems can also be formulated in terms of minimising a function of the form (3.3) or (3.2), by including the penalty that depends on the segment length within the segment cost.

Many authors have looked at different choices of penalties. If we let p denote the number of additional parameters introduced by adding a changepoint, then popular examples used frequently in the literature include $\beta = 2p$ (Akaike's Information Criterion; Akaike, 1974); $\beta = p \log n$ (Schwarz's Information Criterion; Schwarz, 1978); and $\beta = 2p \log \log n$ (Hannan and Quinn, 1979). More sophisticated penalty approaches include the modified Bayesian Information Criterion (mBIC; Zhang and Siegmund, 2007) which accounts for the length of the segments. Whilst these information criteria all have good theoretical properties, they rely on assumptions about the underlying data generating process which gives rise to the data. Unfortunately, in practice there is potential for the modelling assumptions associated with a particular criterion to be violated. Hocking et al. (2013a) show that while the mBIC works well for simulated data-sets where the model assumptions of the mBIC hold, it does not work as well for real data-sets.

An alternative approach is to calculate optimal segmentations with differing num-

bers of changepoints, and then use some alternative method to evaluate each segmentation. This idea has been suggested by Hocking et al. (2013a) who then use annotated training data to learn what choice of penalty is most appropriate for a given application. For recursive methods, such as variants of Binary Segmentation (Scott and Knott, 1974; Fryzlewicz, 2014) and the method of Fryzlewicz (2012), we obtain segmentations corresponding to a range of different numbers of changepoints with no, or little, additional cost. However, if the aim is to find segmentations that are optimal, in terms of minimising a given cost function, these recursive methods cannot be used. Calculating the range of optimal segmentations can be done using the Segment Neighbourhood search algorithm (Auger and Lawrence, 1989), but this comes at a much higher computational cost.

Our contribution is a new algorithm, CROPS, that can compute all optimal segmentations of the penalised minimisation problem as we vary the penalty over some interval. This is similar in spirit to algorithms for variants of penalised regression where, rather than solving a problem for a single penalty value, one calculates the set of solutions obtained as the penalty value varies (Tibshirani and Taylor, 2011; Fryzlewicz, 2012; Zhou and Lange, 2013).

The CROPS algorithm uses a simple relationship between the solutions of the penalised minimisation problem and those of the constrained minimisation problem to find a set of distinct penalty values such that each solution corresponds to either a different segmentation, or will rule out the possibility of an optimal segmentation under the penalised cost with a certain number of changepoints. We show how the computational cost of the method can be improved by storing and reusing certain values that are calculated when solving the penalised cost problem for the earlier choices of the penalty values. The output of CROPS is similar to that of Segment Neighbourhood search, but it can be substantially, even orders of magnitude, faster.

This chapter is organised as follows. In Section 3.2 we introduce the changepoint model and review various ways of detecting multiple changes using both a constrained

and a penalised approach. In Section 3.3 we propose our method for running the detection algorithms over a range of penalty values, and give a bound on its computational cost. Our method will be demonstrated in simulation studies and real data examples in Section 3.4 and Section 3.5.

3.2 Background

3.2.1 Segment Costs

To define the cost of a segmentation we need to specify a segment-specific cost. A common approach, used for example in penalised likelihood (Braun and Müeller, 1998) and minimum description length (Davis et al., 2006) methods, is to introduce a model for the data within a segment. This will define a log-likelihood for the data that depends on a segment-specific parameter. The cost can then be chosen proportional to minus the maximum of this log-likelihood, where we maximise out the segment-specific parameter. The form of this cost will then depend on both modelling assumptions about the distribution of the data-points, and also the type of change that we are attempting to detect. To make this idea concrete, consider the following setting, that we will revisit in the simulation and real-data examples. If we model the data within a segment as being independent and identically distributed, drawn from a Gaussian distribution with mean μ and variance σ^2, then the log-likelihood of the data $y_{(s+1):t}$, up to a common additive constant, would be

$$\ell(y_{(s+1):t}; \mu, \sigma) = -\frac{(t-s)}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=s+1}^{t}(y_j - \mu)^2.$$

For detecting a change in the mean calculating the segment cost involves using minus twice the log-likelihood after maximising over μ. This gives the segment cost:

$$\mathcal{C}(y_{(s+1):t}) = \sum_{j=s+1}^{t}(y_j)^2 - \frac{(\sum_{j=s+1}^{t} y_j)^2}{n}. \tag{3.4}$$

Similarly for detecting a change in both mean and variance, calculating the segment cost would involve using minus twice the log-likelihood after maximising over both μ and σ. This gives a segment cost,

$$\mathcal{C}(y_{(s+1):t}) = (t-s)\left\{\log\left[\frac{1}{t-s}\sum_{j=s+1}^{t}\left(y_j - \frac{1}{t-s}\sum_{i=s+1}^{t} y_i\right)^2\right] + 1\right\}. \tag{3.5}$$

3.2.2 Finding Optimal Segmentations

We now briefly review existing approaches for finding optimal segmentations within the literature.

Segment-Neighbourhood

Auger and Lawrence (1989) introduced the Segment Neighbourhood (SN) search method which is used to solve the constrained problem in (3.1). This method involves specifying the maximum number of changepoints to allow, M, and then calculating the cost of all possible optimal segmentations with 0 to M changepoints. The optimal number of changepoints can then be calculated by (3.2). The computational cost for this method is $\mathcal{O}(Mn^2)$ and thus this method scales poorly when analysing large data-sets with a large number of possible changepoints.

Optimal Partitioning

In order to solve the penalised minimisation problem in (3.3), Jackson et al. (2005) introduced a method also based on dynamic programming: Optimal Partitioning (OP). OP is a recursive process which relates the minimum value of (3.3) to the cost of the optimal segmentation of the data prior to the last changepoint plus the cost of the segment from the last changepoint to the current time-point. For the data up to time s, $y_{1:s}$, we let $\boldsymbol{\tau_s}$ be the set of all possible number and position of changepoints for segmenting the data: $\boldsymbol{\tau_s} = \{\boldsymbol{\tau} : 0 = \tau_0 < \tau_1 < \cdots < \tau_m < \tau_{m+1} = s\}$. If we denote the minimisation of (3.3) for data $y_{1:t}$ by $F(t) = Q(y_{1:t}; \beta)$, with $F(0) = 0$, then this

can be calculated recursively by:

$$F(t) = \min_{\tau \in \boldsymbol{\tau_t}} \left\{ \sum_{i=1}^{m+1} [\mathcal{C}(y_{(\tau_{i-1})+1:\tau_i}) + \beta] \right\} = \min_{s \in \{0,\ldots,t-1\}} \{F(s) + \mathcal{C}(y_{(s+1):t}) + \beta\}. \quad (3.6)$$

This recursion can be interpreted as stating that the minimum cost of segmenting $y_{1:t}$, given the last changepoint is at time s, is the optimal cost for segmenting data up to time s plus the cost of adding a changepoint and the cost for the segment $y_{(s+1):t}$. The value of s which attains the minimum of (3.6) is the position of the last changepoint in the optimal segmentation of $y_{1:t}$. These recursions are solved for $t = 1, 2, ..., n$ with computational cost $\mathcal{O}(n^2)$. Extracting the set of changepoints in the optimal segmentation is achieved by a simple recursion backwards through the data.

3.2.3 Pruning Methods

There has been work on improving these methods through pruning techniques. Killick et al. (2012) introduced a modification of OP; Pruned Exact Linear Time (PELT). This method uses inequality based pruning to remove values of τ which can never be minima from the minimisation performed at each iteration of the OP algorithm. Killick et al. (2012) show that, under certain regularity conditions, the expected computational cost of PELT is $\mathcal{O}(n)$.

Recently Maidstone et al. (2017) proposed an alternative functional based pruning method for OP, FPOP which has an empirical cost of $\mathcal{O}(n)$. This is similar to the pDPA method proposed by Rigaill (2015) who use this functional pruning in Segment Neighbourhood search. pDPA has an empirical cost of $\mathcal{O}(n \log n)$. The disadvantage of both pDPA and FPOP is that they only work in situations where we only have changes in one parameter.

3.3 Algorithm for a Range of Penalty Values

In this section we propose a method which solves the penalised optimisation problem (3.3) for a range of penalty values, β. This method finds the optimal segmentations for a different number of segments without incurring as large a computational cost as solving the constrained optimisation problem for a range of m (the number of changepoints). To achieve this we use a relationship between the penalised and constrained optimisation problems in order to sequentially choose values of β for which the penalised optimisation needs to be solved.

This algorithm can be used within any approach for solving the penalised optimisation problem, which we will define as CPD (as in Change Point Detection) for the remainder of this paper.

3.3.1 Link Between Optimisation Problems

As before, we have $Q_m(y_{1:n})$ as the minimum cost for the constrained optimisation problem (3.1) and $Q(y_{1:n}, \beta)$ as the minimum cost of the penalised optimisation problem (3.3). These costs can be linked by defining the minimum cost for the penalised optimisation problem subject to the number of changepoints being m:

$$P_m(\beta) = Q_m(y_{1:n}) + (m+1)\beta. \tag{3.7}$$

Then we have, for any β,

$$Q(y_{1:n}, \beta) = \min_m P_m(\beta). \tag{3.8}$$

Figure 3.1 shows example $P_m(\beta)$ lines, and the corresponding $Q(y_{1:n}, \beta)$ curve for a range of penalty values, $\beta \in [5.54, 11]$, we discuss interval choice in Section 3.3.3. There are a few important points of interest to note from this plot. Firstly we can clearly see the relationship between the constrained and penalised problems. For

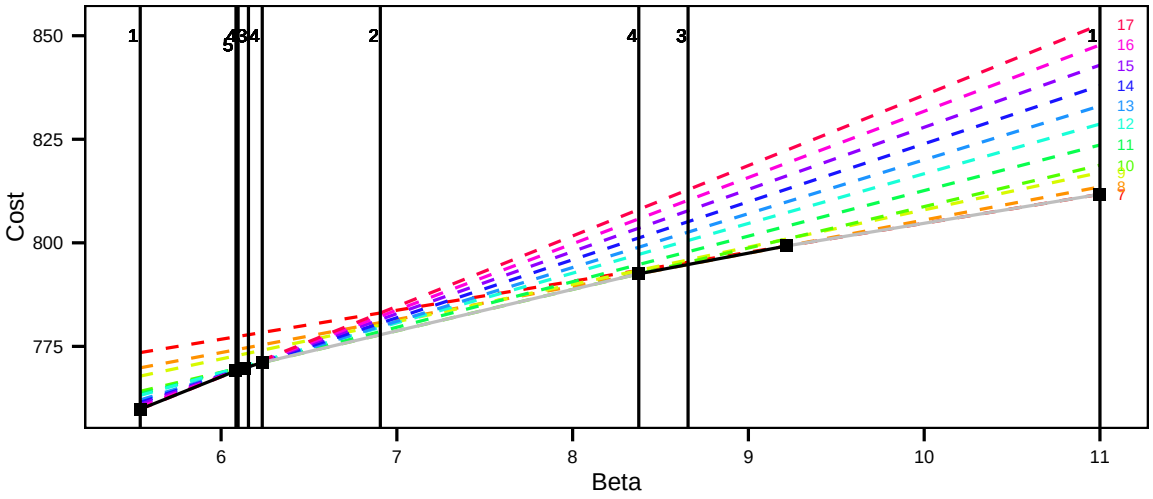


Figure 3.1: Graphical representation of the relationship between the constrained and penalised approaches. The dashed lines are the costs associated with a different number of changepoints plotted against different penalty terms β (3.7). The numbers on the right hand side are the number of changes detected. The solid dark line shows the optimal value of $Q(y_{1:n}, \beta)$ over the range of β. The solid line is split in to 6 subregions highlighted by different shades and the black squares. These indicate the intervals where the optimal number of changepoints is the same for all values of the penalty within the interval. The set of β values for which CPD was run to find all optimal segmentations for $\beta \in [5.54, 11]$ are shown by the vertical lines, interval choice is discussed in Section 3.3.3. The numbers at the top represent the order in which we use the penalty value, note the same numbers represent penalties run in the same step.

example it is evident that using a penalty, $\beta = 10$ and minimising a penalised cost function gives the same optimal segmentation as solving the constrained optimisation problem with $m = 7$. Additionally we can see that as β increases the optimal number of changepoints decreases. By looking at the dashed lines we can see that not all of the possible number of changes are optimal for some β. For our example segmentations with $m = 9, 11, 12, 14$ or 15 are never optimal choices for any β.

Additionally in Figure 3.1 we can see that the penalty values can be partitioned into intervals which all have the same value of m. For instance for all $\beta \in [8.38, 9.22]$ the resulting m is 8. This suggests that if we can learn the boundaries of these intervals, we can use that information to solve the penalised optimisation problem for values of β which will correspond to different optimal segmentations. In particular we

only needed to run CPD for the penalty values indicated on the plot by the vertical lines in order to find all optimal segmentations for $\beta \in [5.54, 11]$. The next Section describes how we find these values of β.

3.3.2 Theoretical Results

We now consider the case where we have solved the penalised optimisation problem for two values of penalty, β_0 and β_1.

For any β we let $m(\beta)$ be the number of changepoints in the segmentation that is optimal for solving the penalised optimisation problem with penalty β. If there is more than one optimal segmentation, we let $m(\beta)$ be the smallest number of changepoints in those optimal segmentations. Note that, trivially, $m(\beta)$ will be a non-increasing function.

Theorem 3.3.1. *Let $\beta_0 < \beta_1$.*

(1) If $m(\beta_0) = m(\beta_1)$ then $m(\beta) = m(\beta_0)$ for all $\beta \in [\beta_0, \beta_1]$.

(2) If $m(\beta_0) = m(\beta_1) + 1$, define

$$\beta_{int} = \frac{Q_{m(\beta_1)}(y_{1:n}) - Q_{m(\beta_0)}(y_{1:n})}{m(\beta_0) - m(\beta_1)}. \tag{3.9}$$

Then $m(\beta) = m(\beta_0)$ if $\beta \in [\beta_0, \beta_{int})$ and $m(\beta) = m(\beta_1)$ if $\beta \in [\beta_{int}, \beta_1]$.

(3) If $m(\beta_0) > m(\beta_1) + 1$, and $m(\beta_{int}) = m(\beta_1)$ where β_{int} is defined by (3.9), then $m(\beta) = m(\beta_0)$ if $\beta \in [\beta_0, \beta_{int})$ and $m(\beta) = m(\beta_1)$ if $\beta \in [\beta_{int}, \beta_1]$.

Proof. See Appendix A. □

3.3.3 The Changepoints for a Range of PenaltieS (CROPS) Algorithm

We now seek to develop a method to find the number of changepoints using different values of the penalty, β, in a range $[\beta_{min}, \beta_{max}]$. Here we introduce the CROPS

algorithm, which sequentially calculates the values of β.

CROPS begins by first running CPD for penalty values β_{min} and β_{max}. Theorem 3.3.1 then shows that if we have $m(\beta_{min}) = m(\beta_{max})$ or $m(\beta_{min}) = m(\beta_{max}) + 1$ we have found all the optimal segmentations for $\beta \in [\beta_{min}, \beta_{max}]$. Otherwise we calculate β_{int} (3.9), the intersection of $P_{m(\beta_{min})}(\beta)$ and $P_{m(\beta_{max})}(\beta)$, then run CPD with this penalty value. By part (3) of Theorem 3.3.1 we know that if $m(\beta_{int}) = m(\beta_{max})$ then we have found all the optimal segmentations for $\beta \in [\beta_{min}, \beta_{max}]$. Otherwise we can now consider the intervals $[\beta_{min}, \beta_{int}]$ and $[\beta_{int}, \beta_{max}]$ separately, and we repeat this procedure on each of those intervals. This continues until there are no new intervals to consider. We are able to use the results above to work out the optimal number of changepoints for all penalty values within the interval $[\beta_{min}, \beta_{max}]$. Pseudo code for this method can be found in Algorithm 1.

Algorithm 1: CROPS algorithm

input : A data-set $y_{1:n} = (y_1, y_2, ..., y_n)$;
Minimum and maximum values of the penalty, β_{min} and β_{max};
`CPD`, an algorithm such as PELT, for solving the penalised optimisation problem.

output: The details of optimal segmentations for each $\beta \in [\beta_{min}, \beta_{max}]$.

1. Run `CPD` for penalty values β_{min} and β_{max};
2. Set $\beta^* = \{[\beta_{min}, \beta_{max}]\}$;

while $\beta^* \neq \emptyset$ **do**
  3. Choose an element of β^*; denote this element as $[\beta_0, \beta_1]$;
  if $m(\beta_0) > m(\beta_1) + 1$ **then**
    4. Calculate $\beta_{int} = \frac{Q_{m(\beta_1)}(y_{1:n}) - Q_{m(\beta_0)}(y_{1:n})}{m(\beta_0) - m(\beta_1)}$.;
    5. Run `CPD` for penalty value β_{int};
    6. **if** $m(\beta_{int}) \neq m(\beta_1)$ **then**
      Set $\beta^* = \{\beta^*, [\beta_0, \beta_{int}), [\beta_{int}, \beta_1]\}$.;
    end
  end
  7. Set $\beta^* = \beta^* \setminus [\beta_0, \beta_1]$;
end

return *Output from running CPD for the set of penalty values.*

Implementing CROPS requires a somewhat arbitrary choice of interval $[\beta_{\min}, \beta_{\max}]$. However it is clearly easier to find an appropriate value of the penalty if we choose

an interval than if we choose just a single value. Furthermore we show in Section 3.3.5 that if our interval appears inappropriate, we can extend the interval at little additional computational cost.

3.3.4 The Number of Changepoints that are Optimal for Some β

For the example in Figure 3.1 we saw some of the optimal segmentations for specific numbers of changepoints would never be optimal regardless of the penalty value used. Thus using this method will not necessarily get the resulting segmentations for all numbers of changepoints, something which you get when you use segment neighbourhood search.

Lavielle (2005) gives a condition under which a segmentation with m changepoints will be the optimal segmentation for some β. Assume that segmentations with $m_1 < \cdots < m_k$ changes, for some $k > 1$, are optimal as we vary $\beta \in [\beta_{min}, \beta_{max}]$. Let $Q_i = Q_{m_i}(y_{1:n})$, for $i = 1, \ldots, k$, be the associated un-penalised cost of these segmentations. We can construct a piece-wise line by joining (m_i, Q_i) with (m_{i+1}, Q_{i+1}) for $i = 1, \ldots, k-1$. All values of changepoints, m, with $m_1 < m < m_k$ and for which there is no optimal segmentation will lie above this line. An example is shown in Figure 3.2.

One way of expressing this condition is that we will not obtain segmentations for which the average reduction in cost of adding some number of changepoints is more than the average increase in cost of removing some number of changepoints. Consider the example in Figure 3.2. By solving the penalised optimisation problem for a range of β we do not find an optimal segmentation with 9 changepoints. This is because the reduction in cost of going from 8 to 9 changepoints is less than for going from 9 to 10 changepoints. It is hard to construct a criteria under which the segmentations not found by solving the penalised optimisation problem would be optimal. In fact Killick et al. (2012) show that any segmentation that is optimal under (3.2) where

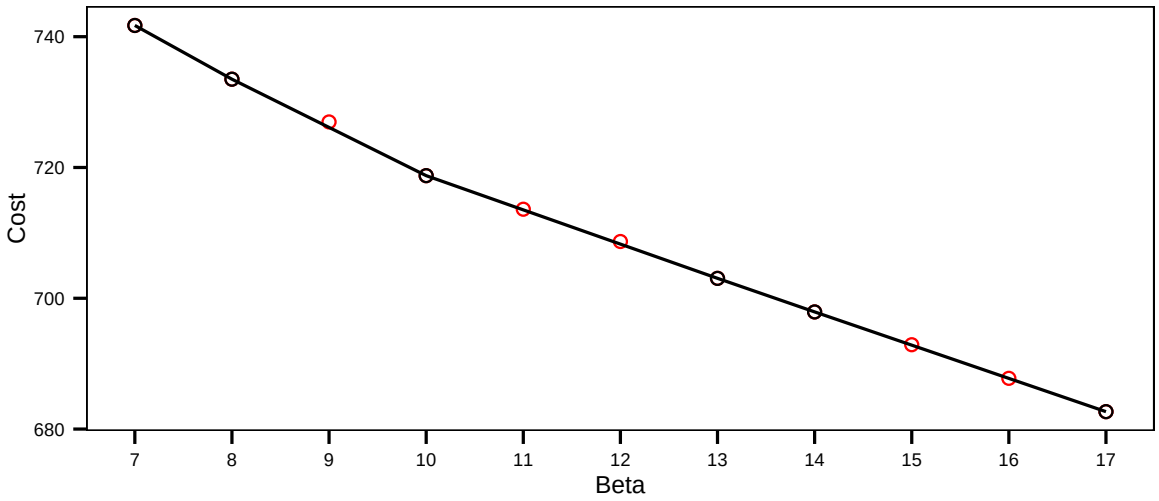


Figure 3.2: Cost for the segmentations against the number of changepoints. The black circles are the points corresponding to optimal segmentations found by solving the penalised optimisation problem over some range of β. The grey circles correspond to the segmentations which are not optimal for any penalty.

the penalty function for adding changepoints, $f(m)$, is concave will be the solution to the penalised optimisation problem for some β.

3.3.5 Computational Cost

We now bound the computational cost of our proposed approach. We do this in terms of the maximum number of times CPD would need to be run. The following theorem shows that this is at most $m(\beta_{min}) - m(\beta_{max}) + 2$ times.

Theorem 3.3.2. *(1) If $m(\beta_0) = m(\beta_1)$ then the maximum number of times that CPD is required to be run to find all the optimal segmentations for $\beta \in [\beta_0, \beta_1]$ is $m(\beta_0) - m(\beta_1) + 2$.*

(2) If $m(\beta_0) > m(\beta_1)$ then the number of times that CPD is required to be run to find all the optimal segmentations for $\beta \in [\beta_0, \beta_1]$ is bounded above by

$$m(\beta_0) - m(\beta_1) + 1.$$

Proof. See Appendix A. □

Often we may choose our interval adaptively. That is we initially choose an interval, $[\beta^{(1)}_{min}, \beta^{(1)}_{max}]$ say. Then, given the set of segmentations we obtain, we may want to increase the upper value of the interval, reduce the lower value, or both. Assume we wish to increase the upper value of the interval (equivalent reasoning applies for reducing the lower value). Denote the new interval of interest by $[\beta^{(1)}_{min}, \beta^{(2)}_{max}]$ with $\beta^{(2)}_{max} > \beta^{(1)}_{max}$. Using the same argument as in the proof of Theorem 3.3.2, the additional cost will be one run of CPD if $m(\beta^{(2)}_{max}) = m(\beta^{(1)}_{max})$, and be at most $m(\beta^{(1)}_{max}) - m(\beta^{(2)}_{max})$ runs otherwise. In the latter case, the overall number of runs of CPD is bounded by $m(\beta^{(1)}_{min}) - m(\beta^{(2)}_{max}) + 2$, which is the same bound as if we used the larger interval initially.

Recycling Calculations

It is possible to speed up Algorithm 1 by recycling some of the calculations for example if in the situation where we use PELT. As we describe in Appendix A, for the PELT algorithm we calculate and store the minimum penalised cost, the number of changepoints in this segmentation for $t = 1, \ldots, n$ and the position of the most recent changepoint up to time t. If PELT was run with penalty value β we denote these values as $F(t, \beta)$, $m(t, \beta)$ and $cp(t, \beta)$ respectively. We can re-use these values from previous runs of PELT to precalculate many of the values for a new run.

Assume we have run PELT with penalty values β_0 and β_1, and are now wanting to run PELT for β_{int} where $\beta_0 < \beta_{int} < \beta_1$. Before running PELT for the new value we iterate for $t = 1, ..., n$:

1. If $m(t, \beta_0) = m(t, \beta_1)$ then set $m(t, \beta_{int}) = m(t, \beta_0)$, $cp(t, \beta_{int}) = cp(t, \beta_0)$ and $F(t, \beta_{int}) = F(t, \beta_0) + m(t, \beta_{int})(\beta_{int} - \beta_0)$.

2. If $m(t, \beta_0) = m(t, \beta_1) + 1$ then calculate $a = F(t, \beta_0) + m(t, \beta_0)(\beta_{int} - \beta_0)$ and $b = F(t, \beta_1) + m(t, \beta_1)(\beta_{int} - \beta_1)$. If $a < b$ then $m(t, \beta_{int}) = m(t, \beta_0)$, $cp(t, \beta_{int}) = cp(t, \beta_0)$ and $F(t, \beta_{int}) = a$; else $m(t, \beta_{int}) = m(t, \beta_1)$, $cp(t, \beta_{int}) = cp(t, \beta_1)$ and $F(t, \beta) = b$.

We then just need to run PELT to calculate the values of $F(t, \beta_{int})$, $m(t, \beta_{int})$ and $cp(t, \beta_{int})$ for times t that we have not been able to precalculate them.

3.4 Simulation Study

This section shows the performance of CROPS in comparison to other methods which find a range of segmentations. In particular we look at two models: the first being a uniform variance model with a change in mean and the second being a model with both changes in mean and variance. For the change in mean model the quickest method for solving the penalised optimisation problem is FPOP (Maidstone et al., 2017, available from https://r-forge.r-project.org/projects/opfp/) and the quickest method for solving the constrained optimisation problem is pDPA (Rigaill, 2015, available in the `Segmentor3IsBack` R package, Cleynen et al. (2013)). Thus we compare the CROPS using FPOP with pDPA. As described in Section 3.2.3 neither pDPA or FPOP can be applied when there is more than one parameter for each segment. So for the change in mean and variance we compare CROPS with PELT against Segment Neighbourhood. In this latter case we also compare the speed of CROPS with and without the recycling of calculations introduced in Section 3.3.5.

Since all of these methods optimise exactly, a solution with m changepoints will have the same m changepoints for all of the methods, we only compare the different methods in terms of speed. We are also able to use CROPS to efficiently study and compare some different proposals for the choice of the penalty. Whilst some of these work well when we use the correct model for the data, we show that they can give misleading results when the model is mis-specified, something that is likely to be a feature of real-life applications of changepoint detection.

3.4.1 Change in Mean

We simulate data of varying lengths with changepoints distributed uniformly in time but with the constraint that there are at least 20 observations between changepoints. For a given value of n we simulate data-sets with a fixed number of changepoints, $m = 2$ (See Appendix A for the cases where we have a linear, $m = n/100$, and sublinear, $m = \sqrt{n}/4$, number of changepoints). We generate the segment means from a normal distribution with mean 0 and standard deviation 2.5 and we let the segment standard deviation be 1. For this model we use the cost function in (3.4).

In the CROPS algorithm we set $\beta_{min} = 4$ and $\beta_{max} = 40$ as indicative values only. For pPDA we set the maximum value of changepoints to be the number of changepoints detected using the smallest value of the penalty value in FPOP. The results are shown in Figure 3.3.

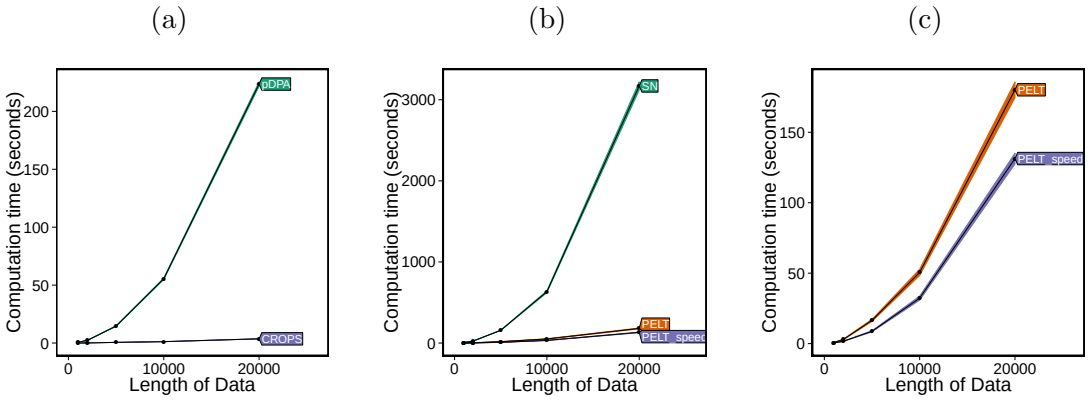


Figure 3.3: (a) CPU cost for using either pDPA or CROPS with FPOP. (b) CPU cost for using SN, CROPS with PELT and CROPS with PELT with the speed improvements. (c) A close up of PELT and PELT with the speed improvements.

It is evident from Figure 3.3a that using CROPS with FPOP is substantially quicker than using pPDPA. As the length of the data-set increases the gains in speed increase.

3.4.2 Change in Mean and Variance

To look at models with a change in mean and variance we simulate data as above but this time we generate the segment means from a normal distribution with mean 0 and standard deviation 2.5, and the segment standard deviations from a Log-normal distribution with mean 0 and standard deviation $\frac{\log(10)}{2}$. In the case with a fixed number of changepoints we use $m = 10$. For this model we use the cost function in (3.5). In the CROPS algorithm we set $\beta_{min} = 14$ and $\beta_{max} = 40$. For SN we set the maximum value of changepoints to be the number of changepoints detected using the smallest value of the penalty value in FPOP.

The results can be seen in Figure 3.3b. Similar to the above results it is evident that CROPS with PELT is much faster than SN. It can be seen in Figure 3.3c that the addition of the recycling of the calculations (PELT_speed) leads to modest gains in speed.

3.4.3 Evaluating the Choice of Penalty

In this section, we use the change in mean and variance model as above and a mis-specified model. For the mis-specified model, for a segment k we simulate segment standard deviations, σ_k^2, and an initial mean value, μ_k. If Y_t is in segment k then we simulate our data from $Y_t \sim N(\nu_t, \sigma_k^2)$, where $\nu_t = \mu_k$ if t is the first point in a segment and $\nu_{t+1} = \nu_t + \epsilon_t$, $\epsilon_t \sim N(0, 0.1)$ otherwise. We generate the initial segment means from a normal distribution with mean 0 and standard deviation 2.5 the segment standard deviations from a Log-normal distribution with mean 0 and standard deviation $\frac{\log(10)}{2}$. The results for the real and mis-specified model are shown in Figures 3.4 and 3.5 respectively. To evaluate the penalty choice we initially find the range of β values which estimate the correct number of changepoints. For a given simulation scenario ($n = 10,000$) we calculate the average of this range over 100 simulated data-sets, and compare this average with the different penalty choices (Figures 3.4a and 3.5a). We then look at the proportion of true positive changepoints, $\xi(\mathbf{C}||\hat{\mathbf{C}})$, and false posi-

tive changepoints, $\xi(\hat{\mathbf{C}}||\mathbf{C})$ (Figures 3.4b and 3.5b). To calculate these we define an actual changepoint as detected if we infer a changepoint within 10 time-points of its location. Let $\mathbf{C}$ be the vector of $n_{\mathbf{C}}$ true changepoint positions, and $\hat{\mathbf{C}}$ be the vector of $n_{\hat{\mathbf{C}}}$ estimated changepoint positions. If $\mathbb{1}(\cdot)$ is an indicator function then

$$\xi(\mathbf{C}||\hat{\mathbf{C}}) = \frac{\sum_{i=1}^{n_{\mathbf{C}}} \mathbb{1}\left(\min_j\{|C_i - \hat{C}_j|\} \leq 10\right)}{n_{\mathbf{C}}} \text{ and } \xi(\hat{\mathbf{C}}||\mathbf{C}) = 1 - \frac{n_{\mathbf{C}}\xi(\mathbf{C}||\hat{\mathbf{C}})}{n_{\hat{\mathbf{C}}}}. \quad (3.10)$$

For the real model case we can also look at the mean square error (MSE) to evaluate the accuracy of estimates of the segment parameters. That is if $\hat{\theta}_i$ is an estimated parameter of the observation at time i, and θ_i the true parameter then MSE is $\sum_{i=1}^{n}(\hat{\theta}_i - \theta_i)^2/n$. We look at MSE for the mean and standard deviation separately (Figure 3.4c).

From the real model results it can be seen that, in this example, when we have 10 changepoints in the data the optimal value of the penalty lies in a wide interval which increases with data size. In this case we can see that the AIC, SIC and Hannan-Quinn penalty values will all over-fit the data. From further simulations (see Appendix A) we found that when the number of changepoints increases with the amount of data, the interval in which the optimal penalty value lies decreases as the length of the data increases. In this case the SIC underestimates the number of changes whereas the AIC and Hannan-Quinn penalty term both overestimate the number of changes. When there is a sublinear number of changepoints the optimal penalty value lies in a smaller interval than it did when there was a fixed number of changes. In this case the SIC, AIC and Hannan-Quinn penalty all overestimate the number of changepoints.

In terms of accuracy it is clear to see that both the AIC and Hannan-Quinn penalty detect a lot of false positive changepoints. The SIC penalty outperforms the Hannan-Quinn penalty for estimating the segment parameters. In all cases the MSE for the AIC penalty term was much larger than the other two penalties and thus not shown.

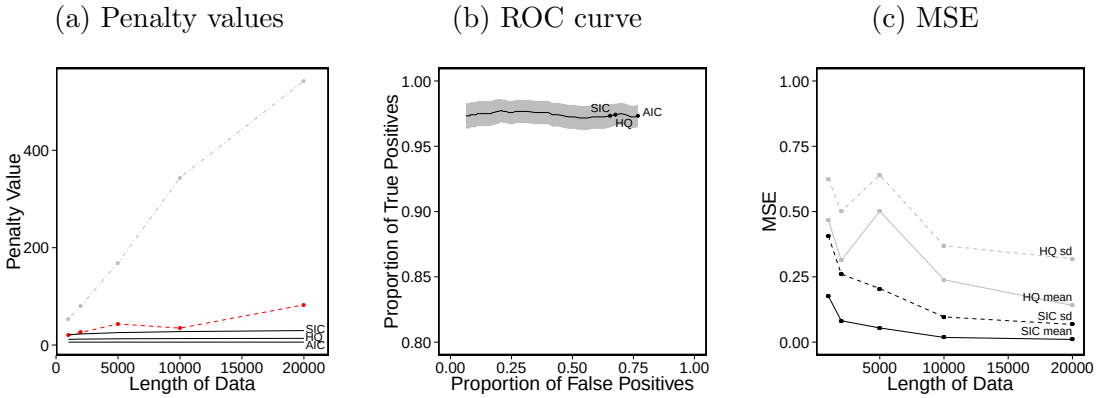


Figure 3.4: Results for the true model. (a) Average minimum (black, dashed) and maximum (grey, dot-dashed) optimal penalty values in comparison to popular penalty terms in the literature. Solid lines from top to bottom are the SIC, Hannan-Quinn and AIC penalty values. (b) Proportion of true positives against the proportion of false positives for $n = 10,000$. (c) MSE for the mean (solid) and the standard deviation (dashed).

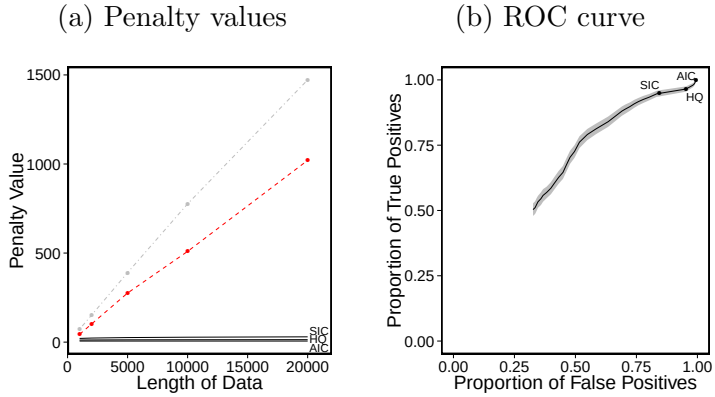


Figure 3.5: Results for the mis-specified model scenario. (a) Average minimum (black, dashed) and maximum (grey, dot-dashed) optimal penalty values in comparison to popular penalty terms in the literature. (b) Proportion of true positives against the proportion of false positives for $n = 10,000$.

We now look at the case where we have the mis-specified model. These results can be seen in Figure 3.5. It is obvious from these results that the optimal penalty value, in terms of correctly estimating the number of changepoints, is much greater than that for the correctly specified model. It is also much larger than any of SIC, AIC and Hannan-Quinn. From the accuracy plot we can see that none of the penalty

terms perform well, with them all detecting a large number of false positives.

3.5 Application to Hi-C Data

Lévy-Leduc et al. (2014) look at detecting genomic regions that interact through the folding and 3-D structure of the chromosome. They achieve this through changepoint detection from a deep sequencing approach called Hi-C. A chromosome is split into a series of *windows* of consecutive base-pairs on a chromosome. The Hi-C data consists of measurements, y_{ij}, of the amount of interaction between window i and window j. We expect regions that interact to be contiguous along a chromosome, so the windows are ordered based on position along the chromosome. Then Lévy-Leduc et al. (2014) segment the data into $m+1$ contiguous regions, where $\mathrm{E}(Y_{ij}) = \mu_s$ if gene i and j are both in segment s for some $s = 1, \ldots, m+1$; and $\mathrm{E}(Y_{ij}) = \mu_0 \approx 0$ if genes i and j are in different segments. Example data from the first 200 windows on chromosome 16 is shown in Figure 3.6(a). Note that there is a single measurement for each pair of windows, so we have set $y_{ij} = y_{ji}$. A segmentation produces square regions on the diagonal of the data matrix, corresponding to the measurements between pairs of genes that have been grouped together.

Lévy-Leduc et al. (2014) formulate the segmentation problem in the form of minimising the penalised cost $\sum_{i=1}^{m+1} \mathcal{C}(y_{(\tau_{i-1}+1):\tau_i}) + \beta m$. They consider a range of different segment costs. We will focus on one, where $\mathcal{C}(y_{s:t})$ is defined in terms of data y_{ij} with $s \leq i \leq t$ and $j < i$, by

$$\mathcal{C}(y_{s:t}) = \min_{\mu} \left[\sum_{i=s+1}^{t} \sum_{j=s}^{i-1} (y_{ij} - \mu)^2 \right] + \min_{\mu_0} \left[\sum_{i=s}^{t} \sum_{j=1}^{s-1} (y_{ij} - \hat{\mu}_0)^2 \right].$$

This is a non-standard segmentation problem. Brault et al. (2015) show that, under a form of in-fill asymptotics, you can consistently estimate the number of changepoints using $\beta = 0$, and this is the choice used in Lévy-Leduc et al. (2014). We will

consider using CROPS to study segmentations for a range of penalty values. Note that for this application, the cost function does not satisfy the condition explained in Killick et al. (2012) for PELT, since adding a change does not necessarily reduce the cost and thus we use Optimal Partitioning.

Figure 3.6 shows the results for analysing data from chromosome 16 (in total over 2.4 million data-points corresponding to 2,221 windows). We ran CROPS with the interval $[0, 1000]$ which required us to solve the Optimal Partitioning recursions just 34 times, whereas Segment Neighbourhood would have required us to solve an almost identical set of recursions 217 times; thus CROPS reduces the computational cost of the dynamic programming recursions by an order of magnitude. In order to then pick the best segmentation we use a method suggested by Lavielle (2005), which looks at how the minimum value of the cost changes as we add more changepoints. To do this we plot the un-penalised cost against the number of segments, m (Figure 3.6b). Initially as we increase m we are likely to be detecting true changes, these will eventually become false positives, and we would expect that detecting a false positive will not lower the cost as much. Thus Lavielle (2005) suggests choosing the point where the decrease in cost due to detecting a further changepoint noticeably changes. This can be thought of as looking for an "elbow" in the plot. In practice such an approach may suggest a plausible range of values for m and these could then be considered in turn as alternative segmentations.

Using this approach suggests a segmentation with 200 changepoints. By comparison using $\beta = 0$, as suggested by Lévy-Leduc et al. (2014) finds 217 changepoints, and using the SIC penalty produces a segmentation with 214 changepoints. In Figures 3.6c and 3.6d we plot two regions where there was greatest disparity, between the three segmentations. Plots of other regions with differences in segmentations are in the online supplementary material. The segmentations with the SIC penalty or with $\beta = 0$ seem to over-fit the data, introducing changepoints into regions, such as around window 380 or window 560, where there is little signal.

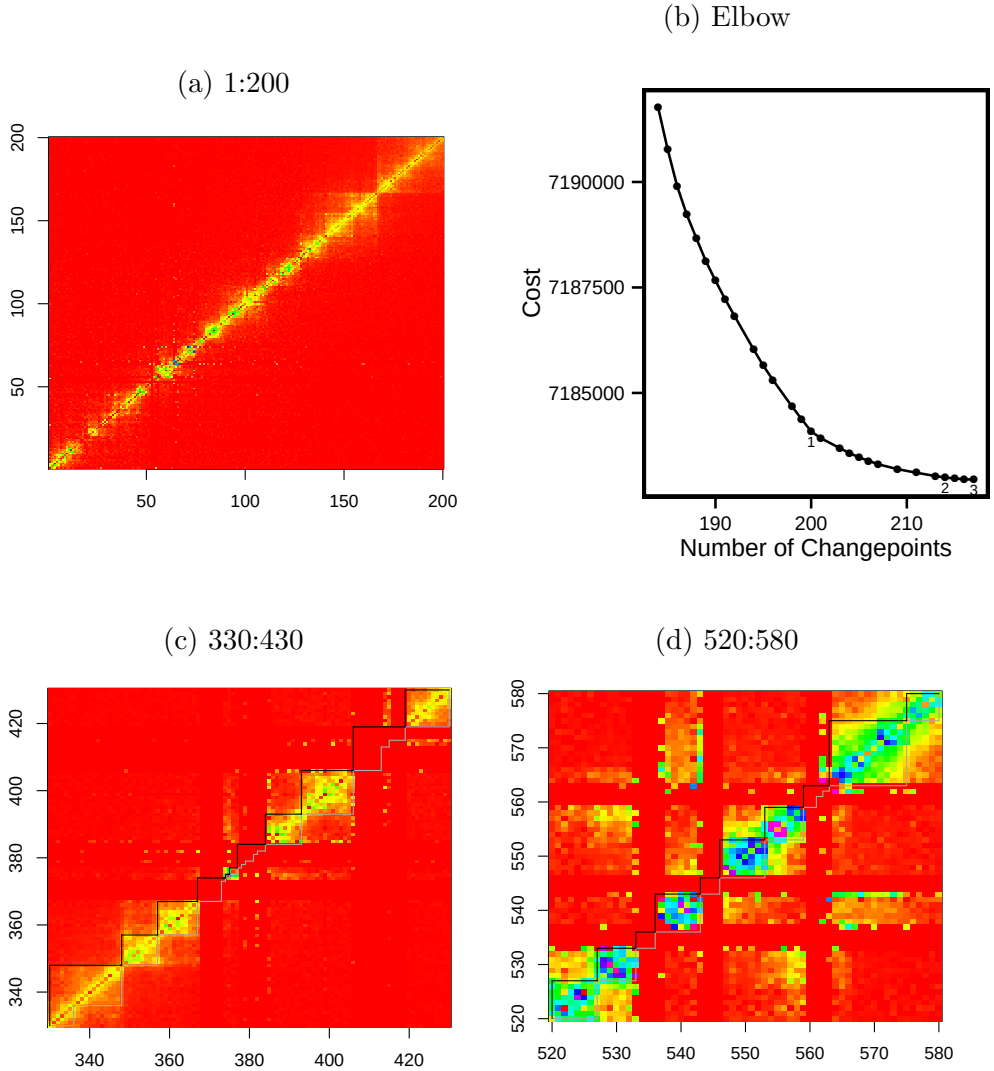


Figure 3.6: (a) First 200 × 200 data-points of chromosome 16. (b) The costs vs number of changepoints for chromosome 16 where 1 is the point we use as being on the "elbow" and thus refer this to the optimal penalty value, 2 is when we use the SIC penalty and 3 is when the penalty is equal to 0. (c) Close up of the segmentations for windows 330 to 430. (d) close up of the segmentations for windows 520 to 580. In both cases the black line is our segmentation and the grey line is the segmentation using $\beta = 0$. For (c) using SIC gives the same segmentation as $\beta = 0$; whilst for (d) using SIC gives the same segmentation as ours.

3.6 Discussion

In this chapter we have developed a method, CROPS, to obtain the optimal segmentations of data, based on minimising a penalised cost function, for a range of penalty values. For many applications, we believe this is a more appropriate approach to segmenting data than just using a single choice of penalty, such as SIC. In particular, whilst default choices can work well if we have an accurate model for the data within each segment, we have shown that they lack robustness, and can produce poor segmentations, in the presence of model mis-specification. We have observed such issues in both a simulation study, and when analysing the genome data.

Minimising the penalised cost function for a range of penalty values is one way of producing a number of different ways of segmenting data, each with a different number of segments. As such, this approach is an alternative to the Segment Neighbourhood search method (and the corresponding pruned method, pDPA), which outputs the optimal segmentation as the number of segments is varied across a suitably chosen range. The advantage of the new approach is one of computational speed, which benefits from the fact that minimising the penalised cost function is a simpler problem to solve than minimising the cost function under a constraint on the number of changepoints, the problem that Segment Neighbourhood solves. In our simulations, CROPS was up to two orders of magnitude quicker than Segment Neighbourhood. One advantage of Segment Neighbourhood is that it produces an optimal segmentation for all numbers of segments in the chosen range, whereas some of these may not be optimal under the penalised cost function for any penalty value, and hence not found via our new method. However the segmentations we do not recover correspond to, for example, ones where adding an extra changepoint leads to a larger change in cost than removing a changepoint. It is hard to construct a sensible criteria under which such segmentations would be optimal.

Code implementing CROPS is available in the R changepoint package, (Killick et al., 2014).

Chapter 4

A Computationally Efficient Nonparametric Approach for Changepoint Detection

4.1 Introduction

Changepoint detection is an area of statistics broadly studied across many disciplines such as acoustics (Guarnaccia et al., 2015; Lu and Zhang, 2002), genomics (Olshen et al., 2004; Zhang and Siegmund, 2007) and oceanography (Nam et al., 2015). Whilst the changepoint literature is vast, many existing methods are parametric. For example a common approach is to introduce a model for the data within a segment, use minus the maximum of the resulting log-likelihood to define a cost for a segment, and then define a cost of a segmentation as the sum of the costs for each of its segments. See for example Yao (1988); Lavielle (2005); Killick et al. (2012); Davis et al. (2006). Finally, the segmentation of the data is obtained as the one that minimises a penalised version of this cost (see also Frick et al., 2014, for an extension of these approaches).

A second class of methods are based on tests for a single changepoint, with the tests often defined based on the type of change that is expected (such as change in

mean), and the distribution of the null-statistic for each test depending on further modelling assumptions for the data (see e.g. Bai and Perron, 1998; Dette and Wied, 2016). Tests for detecting a single change can then be applied recursively to detect multiple changes, for example using Binary Segmentation (Scott and Knott, 1974) or its variants (e.g. Fryzlewicz, 2014). For a review of alternative approaches for change detection see Jandhyala et al. (2013) and Aue and Horváth (2013).

Much of the existing literature on nonparametric methods look at single changepoint detection (Page, 1954; Bhattacharyya and Johnson, 1968; Carlstein, 1988; Dümbgen, 1991). Several approaches are based on using rank statistics such as the Mann-Whitney test statistic (Pettitt, 1979). Ross and Adams (2012) introduce the idea of using the Kolmogorov-Smirnov and the Cramer-von Mises test statistics; both of which use the empirical distribution function. Other methods include using kernel density estimations (Baron, 2000), however these can be computationally expensive to calculate.

There is less literature on the nonparametric multiple changepoint setting. The single changepoint detection methods which have been developed using nonparametric methods do not extend easily to multiple changepoints. Within the sequential changepoint detection literature one can treat the problem as a single changepoint problem which resets every time a changepoint is detected (Ross and Adams, 2012). Lee (1996) proposed a weighted empirical measure which is simple to use but has been shown to have unsatisfactory results. Under the multivariate setting Matteson and James (2014) and James and Matteson (2015) proposed methods, E-divisive and e-cp3o, based on clustering and probabilistic pruning respectively. The E-divisive method uses an exact test statistic with an approximate search algorithm whereas the e-cp3o method uses an approximate test statistic with an exact search algorithm. As a result e-cp3o is faster but lacks slightly in the quality for the changepoints detected.

In this chapter we focus on univariate changepoint detection and we are interested in the work of Zou et al. (2014) who propose a nonparametric likelihood based on the

empirical distribution. They then use a dynamic programming approach, Segment Neighbourhood Search (Auger and Lawrence, 1989), which is an exact search procedure, to find multiple changepoints. Whilst this method is shown to perform well, it has a computational cost of $\mathcal{O}(Mn^2 + n^3)$ where M is the maximum number of changepoints and n is the length of the data. This makes this method infeasible when we have large data-sets, particularly in situations where the number of changepoints increases with n. To overcome this, Zou et al. (2014) propose an additional screening step that prunes many possible changepoint locations. However, as we establish in this chapter, this screening step can adversely affect the accuracy of the final inferred segmentation.

In this chapter we seek to develop a computationally efficient approach to the multiple changepoint search problem in the nonparametric setting. Our approach is an extension to the method of Zou et al. (2014), which uses the cumulative empirical distribution function to define segment costs. Our method firstly involves simplifying the definition of the segment cost, so that calculating the cost for a given segment involves computation that is $O(\log n)$ rather than $O(n)$. Secondly we apply a different dynamic programming approach, Pruned Exact Linear Time (PELT) (Killick et al., 2012), that is substantially quicker than Segment Neighbourhood Search; for many situations where the number of changepoints increases linearly with n, PELT has been proven to have a computational cost that is linear in n.

We call the new algorithm ED-PELT, referring to the fact we have adapted PELT with a cost function based on the empirical distribution. A disadvantage of ED-PELT is that it requires the user to pre-specify a value by which the addition of a changepoint is penalised. The quality of the final segmentation can be sensitive to this choice, and whilst there are default choices these do not always work well. However we show that the Changepoints for a Range of PenaltieS (CROPS) algorithm (Haynes et al., 2017) can be used with ED-PELT to explore optimal segmentations for a range of penalties.

The rest of this chapter is organised as follows. In Section 4.2 we give details

of the NMCD approach proposed by Zou et al. (2014). In Section 4.3 we introduce our new efficient nonparametric search approach, ED-PELT, and show how we can substantially improve the computational cost of this method. In Section 4 we demonstrate the performance of our method on simulated data-sets comparing our method with NMCD. Finally in Section 5 we include some simulations which analyse the performance of NMCD for different scenarios and then we show how a nonparametric cost function can be beneficial in situations where we do not know the underlying distribution of the data. In order to demonstrate our method we use heart-rate data recorded whilst an individual is running.

4.2 Nonparametric Changepoint Detection

4.2.1 Model

The model that we refer to throughout this paper is as follows (note we have made a slight change in notation to that introduced in Section 2.2 and used in Chapter 3 to highlight the fact that the data is nonparametric). Assume that we have data, $x_1, ..., x_n \in \mathbb{R}$, that have been ordered based on some covariate information such as time or position along a chromosome. For $v \geq u$ we denote $x_{u:v} = \{x_u, ..., x_v\}$. Throughout we let m be the number of changepoints, and the positions be $\tau_1, \ldots, \tau_m$. Furthermore we assume that τ_i is an integer and that $0 = \tau_0 < \tau_1 < \tau_2 < ... < \tau_m < \tau_{m+1} = n$. Thus our m changepoints split the data into $m + 1$ segments, with the ith segment containing $x_{\tau_{i-1}+1:\tau_i}$

As in Zou et al. (2014) we will let $F_i(t)$ be the (unknown) cumulative distribution function (CDF) for the ith segment, and $\hat{F}_i(t)$ the empirical CDF. In other words

$$\hat{F}_i(t) = \frac{1}{\tau_i - \tau_{i-1}} \times \left(\sum_{j=\tau_{i-1}+1}^{\tau_i} \mathbb{1}\{x_j < t\} + 0.5 \times \mathbb{1}\{x_j = t\} \right). \tag{4.1}$$

Finally we let $\hat{F}(t)$ be the empirical CDF for the full data-set. Here the $0.5 \times \mathbb{1}\{x_j = t\}$ term shares $x_j = t$ equally in both $\mathbb{1}\{x_j \leq t\}$ and $\mathbb{1}\{x_j \geq t\}$. Normally in changepoint detection we minimise an optimisation problem however if we instead maximised the optimisation problem the $0.5 \times \mathbb{1}\{x_j = t\}$ will ensures that we will get the same results as had we minimised.

4.2.2 Nonparametric Maximum Likelihood

If we have n data-points that are independent and identically distributed with CDF $F(t)$, then, for a fixed value of t, the empirical CDF will satisfy $n\hat{F}(t) \sim \text{Binomial}(n, F(t))$. Hence the log-likelihood of $F(t)$ is given by: $n\{\hat{F}(t)\log(F(t)) + (1 - \hat{F}(t))\log(1 - F(t))\}$. This log-likelihood is maximised by the value of the empirical CDF, $\hat{F}(t)$. We can thus use minus the maximum value of this log-likelihood as a segment cost function. So for segment i we have a cost that is $-\mathcal{L}_{np}(x_{(\tau_{i-1}+1):\tau_i}|t)$ where

$$\mathcal{L}_{np}(x_{(\tau_{i-1}+1):\tau_i}|t) = (\tau_i - \tau_{i-1}) \times [\hat{F}_i(t)\log\hat{F}_i(t) + (1 - \hat{F}_i(t))\log(1 - \hat{F}_i(t))]. \quad (4.2)$$

We can then define a cost of a segmentation as the sum of the segment costs. Thus to segment the data with m changepoints we minimise $-\sum_{i=1}^{m+1}\mathcal{L}_{np}(x_{(\tau_{i-1}+1):\tau_i}|t)$.

4.2.3 Nonparametric Multiple Changepoint Detection

One problem with the segment cost as defined by (4.2) is that it only uses information about the CDF evaluated at one value of t and that the choice of t can have detrimental effects on the resulting segmentations. To overcome this Zou et al. (2014) suggest defining a segment cost which integrates (4.2) over different values of t. They suggest a cost function for a segment with data $x_{u:v}$ that is

$$\int_{-\infty}^{\infty} -\mathcal{L}_{np}(x_{u:v}|t)dw(t), \quad (4.3)$$

with a weight, $dw(t) = \{F(t)(1-F(t))\}^{-1}dF(t)$, that depends on the CDF of the full data. This weight is chosen to produce a powerful goodness of fit test (Zhang, 2002). As this is unknown they approximate it by the empirical CDF of the full data, and then further approximate the integral by a sum over the data-points. This gives the following objective function

$$Q_{\text{NMCD}}(\tau_{1:m}|x_{1:n}) = -n\sum_{i=1}^{m+1}\sum_{t=1}^{n}(\tau_i - \tau_{i-1}) \times \frac{\hat{F}_i(t)\log\hat{F}_i(t) + (1-\hat{F}_i(t))\log(1-\hat{F}_i(t))}{(t-0.5)(n-t+0.5)}. \tag{4.4}$$

For a fixed m this objective function is minimised to find the optimal segmentation of the data.

In practice a suitable choice of m is unknown, and Zou et al. (2014) suggest estimating m using the Bayesian Information criterion (Schwarz, 1978). That is, they minimise

$$\text{BIC} = \min_{m|\tau_1,\ldots,\tau_m}\left\{Q_{\text{NMCD}}(\tau_{1:m}|x_{1:n}) + m\xi_n\right\}, \tag{4.5}$$

where ξ_n is a sequence going to infinity.

4.2.4 NMCD Algorithm

To maximise the objective function (4.4), Zou et al. (2014) use the dynamic programming algorithm Segment Neighbourhood Search (Auger and Lawrence, 1989). This algorithm calculates the optimal segmentations, given a cost function, for each value of $m = 1, \ldots, M$, where M is a specified maximum number of changepoints to search for. If all the segment costs have been pre-computed then Segment Neighbourhood search has a computational cost of $\mathcal{O}(Mn^2)$. However for NMCD the segment cost

involves calculating

$$\sum_{t=1}^{n} \frac{\hat{F}_i(t)\log\hat{F}_i(t) + (1-\hat{F}_i(t))\log(1-\hat{F}_i(t))}{(t-0.5)(n-t+0.5)},$$

and thus calculating the cost for a single segment is $O(n)$. Hence the cost of pre-computing all segment costs is $O(n^3)$, and the resulting algorithm has a cost that is $O(Mn^2 + n^3)$.

To reduce the computational burden when we have long data-series, Zou et al. (2014) propose a screening step. They consider overlapping windows of length $2N_I$ for some $N_I \in \mathbb{R}$. For each window they calculate the Cramér-von Mises (CvM) statistic for a changepoint at the centre of the window. They then compare these CvM statistics, each corresponding to a different changepoint location, and remove a location as a candidate changepoint if its CvM statistic is smaller than any of the CvM statistics for locations within N_I of it. The number of remaining candidate changepoint positions is normally much smaller than n and thus the computational complexity can be substantially reduced. The choice of N_I is obviously important, with larger values leading to the removal of more putative changepoint locations, but at the risk or removing true changepoint locations. In particular, the rationale for the method is based on N_I being smaller than any segment that you wish to detect. As a default, Zou et al. (2014) recommend choosing $N_I = \lceil (\log n)^{3/2}/2 \rceil$ where $\lceil x \rceil$ denotes the smallest integer which is larger than x.

4.3 ED-PELT

Here we develop a new, computationally efficient, way to segment data using a cost function based on (4.3). This involves firstly an alternative numerical approximation to the integral (4.3), which is more efficient to calculate. In addition we use a more efficient dynamic programming algorithm, PELT (Killick et al., 2012), to then minimise the cost function.

4.3.1 Discrete Approximation

To reduce the cost of calculating the segment cost, we approximate the integral by a sum with $K << n$ terms. The integral in (4.3) involves a weight, and we first make a change of variables to remove this weight.

Lemma 4.3.1. *Let* $c = -\log(2n-1)$. *For* $z \in [-1,1]$ *define* $p(z) = (1+\exp\{cz\})^{-1}$. *Then*

$$\int_{\frac{1}{2n}}^{\frac{2n-1}{2n}} \mathcal{L}_{np}(x_{u:v}|t)\{F(t)(1-F(t))\}^{-1}dF(t) = -c\int_{-1}^{1} \mathcal{L}_{np}(x_{u:v}|F^{-1}(p(z)))dz. \quad (4.6)$$

Proof. This follows from making the change of variable $F(t) = p(z)$. □

Using Lemma 4.3.1, we suggest the following approximation, based on an approximation of (4.6) using K unevenly spaced x-values. We choose these x-values specifically to give higher weight to values in the tail of the distribution of the data. Our approximation achieves this through a sum where each term has equal weight, but where the x-values we choose are preferentially chosen from the tail of the distribution. That is we fix K, and let $t_1, \ldots, t_K$ be such that t_k is the $(1+(2n-1)\exp\{\frac{c}{K}(2k-1)\})^{-1}$ empirical quantile of the data, where c is defined in Lemma 4.3.1. then we approximate (4.3) by

$$\mathcal{C}_K(x_{u:v}) = \frac{-2c}{K}\sum_{k=1}^{K} \mathcal{L}_{np}(x_{u:v}|t_k). \quad (4.7)$$

The cost now for calculating the segment costs is $\mathcal{O}(K)$. We show empirically in Section 4.4 that this choice of K can lead to segment costs of $\mathcal{O}(\log n)$.

4.3.2 Use of PELT

We now turn to consider how the PELT approach of Killick et al. (2012) can be incorporated within this framework. The PELT dynamic programming algorithm is

able to solve minimisation problems of the form

$$Q_{\mathrm{PELT}}(x_{1:n}|\xi_n) = \min_{m,\tau_{1:m}} \left\{ \sum_{i=1}^{m+1} [\mathcal{C}_K(x_{(\tau_{i-1}+1):\tau_i}) + \xi_n] \right\}.$$

It jointly minimises over both the number and position of the changepoints, but requires the prior choice of ξ_n, the penalty value for adding a changepoint. The PELT algorithm uses the fact that $Q_{\mathrm{PELT}}(x_{1:n})$ is the solution of the recursion, for $v > 1$

$$Q_{\mathrm{PELT}}(x_{1:v}|\xi_n) = \min_{u<v} \left(Q_{\mathrm{PELT}}(x_{1:u}) + \mathcal{C}_K(x_{u+1:v}) + \xi_n \right). \tag{4.8}$$

The interpretation of this is that the term in the brackets on the right-hand side of (4.8) is the cost for segmenting $x_{1:v}$ with the most recent changepoint at u. We then optimise over the location of this most recent changepoint. Solving the resulting set of recursions leads to an $O(n^2)$ algorithm (Jackson et al., 2005), as (4.8) needs to be solved for $v = 2, \ldots, n$; and solving (4.8) for a given value of v involves a minimisation over v terms.

The idea of PELT is that we can substantially speed up solving (4.8) for a given v by reducing the set of values of u we have to minimise over. This can be done through a simple rule that enables us to detect time points u which can never be the optimal location of the most recent changepoint at any subsequent time. For our application this comes from the following result

Theorem 4.3.2. *If at time v, we have $u < v$ such that*

$$Q_{PELT}(x_{1:u}|\xi_n) + \mathcal{C}_K(x_{u+1:v}) \geq Q_{PELT}(x_{1:v}|\xi_n), \tag{4.9}$$

then for any future time $T > v$, u can never be the time of the optimal last changepoint prior to T.

Proof. This follows from Theorem 3.1 of Killick et al. (2012), providing we can show

that for any $u < v < T$

$$\mathcal{C}_K(x_{u+1:T}) \geq \mathcal{C}_K(x_{u+1:v}) + \mathcal{C}_K(x_{v+1:T}). \tag{4.10}$$

As $\mathcal{C}_K(\cdot)$ is a sum of k terms, each of the form $-\mathcal{L}_{np}(\cdot|t_k)$ we need only show that for any t

$$\mathcal{L}_{np}(x_{(u+1):T}|t) \leq \mathcal{L}_{np}(x_{(u+1):v}|t) + \mathcal{L}_{np}(x_{(v+1):T}|t).$$

Now if we introduce notation that $\hat{F}_{u,v}(t)$ is the empirical CDF for data $x_{u:v}$, we have

$$\begin{aligned}
\mathcal{L}_{np}(x_{(u+1):T}|t) &= (T-u)[\hat{F}_{u,T}(t)\log(\hat{F}_{u,T}(t)) + (1-\hat{F}_{u,T}(t))\log(1-\hat{F}_{u,T}(t))] \\
&= \{(v-u)[\hat{F}_{u,v}(t)\log(\hat{F}_{u,T}(t)) + (1-\hat{F}_{u,v}(t))\log(1-\hat{F}_{u,T}(t))] \\
&+ (T-v)[\hat{F}_{v,T}(t)\log(\hat{F}_{u,T}(t)) + (1-\hat{F}_{v,T}(t))\log(1-\hat{F}_{u,T}(t))]\} \\
&\leq \mathcal{L}_{np}(x_{(u+1):v}|t) + \mathcal{L}_{np}(x_{(v+1):T}|t),
\end{aligned}$$

as required. □

Thus at each time-point we can check whether (4.9) holds, and if so prune time-point u. Under certain regularity conditions, Killick et al. (2012) show that for models where the number of changepoints increases linearly with n, such substantial pruning occurs that the PELT algorithm will have an expected computational cost that is $O(n)$. We call the resulting algorithm we obtain ED-PELT (PELT with a cost based on the empirical distribution).

4.4 Results

4.4.1 Performance of NMCD

We firstly compare the NMCD algorithm with (NMCD+) and without screening (NMCD) using the `nmcdr` R package (Zou and Zhange (2014)), with the default

choices ξ_n (Bayesian Information Criterion) and in the NMCD+ algorithm N_I as detailed in Section 4.2.4. We set up a similar simulation as in Zou et al. (2014). That is, we simulate data of length $n = 1000$ from the following three models, where $J(x) = \{1 + sgn(x)\}/2$.

Model 1: $x_i = \sum_{j=1}^{M} h_j J(nt_i - \tau_j) + \sigma\xi_i$, where

$$\{\tau_j/n\} = \{0.1, 0.13, 0.15, 0.23, 0.25, 0.40, 0.44, 0.65, 0.76, 0.78, 0.81\},$$
$$\{h_j\} = \{2.01, -2.51, 1.51, -2.01, 2.51, -2.11, 1.05, 2.16, -1.56, 2.56, -2.11\},$$

and there are n equally spaced t_i in $[0, 1]$.

Model 2: $x_i = \sum_{j=1}^{M} h_i J(nt_i - \tau_j) + \sigma\xi_i \prod_{j=1}^{\sum_{j=1}^{M} J(nt_i - \tau_j)} v_j$, where

$$\{\tau_j/n\} = \{0.20, 0.40, 0.65, 0.85\}, \{h_j\} = \{3, 0, -2, 0\}, \text{ and } \{v_j\} = \{1, 5, 1, 0.25\}.$$

Model 3: $x_i \sim F_j(x)$, where $\tau_j/n = \{0.20, 0.50, 0.75\}$, $j = 1, 2, 3, 4$, and $F_1(x), ..., F_4(x)$ corresponds to the standard normal, the standardized $\chi^2_{(3)}$ (with zero mean and unit variance), the standardized $\chi^2_{(1)}$ and the standard normal distribution respectively.

The first model has $M = 11$ changepoints, all of which are changes in location. Model 2 has both changes in location and in scale and model 3 has changes in skewness and in kurtosis. For the first two models we also consider three distributions for the error, ξ_i: $N(0, 1)$, Student's t distribution with 3 degrees of freedom and the standardised chi-square distribution with one degree of freedom, $\chi^2_{(1)}$.

To compare both the NMCD and NMCD+ we first look at the true and false discovery rates. That is a detected changepoint $\hat{\tau}_i$ is true if $\min_{1 \le j \le m}\{|\hat{\tau}_i - \tau_j|\} \le h$, where m is the true number of changepoints and h is some threshold. In this case we will use $h = 0$. That is a detected changepoint is only counted as true if it is in the correct location. The number of true detected changepoints is thus $\hat{m}_{\text{TRUE}} =$

$\sum_{i=1}^{\hat{m}} \mathbb{1}_{\min_{1\leq j\leq m}\{|\hat{\tau}_i-\tau_j|\}\leq 0}$, where $\hat{m}$ is the number of detected changepoints. The true discovery rate (TDR) and false discovery rate (FDR) are then calculated as:

$$\text{TDR} = \frac{\hat{m}_{\text{TRUE}}}{m}, \quad \text{FDR} = \frac{\hat{m}_{\text{FALSE}}}{\hat{m}} = \frac{1-\hat{m}_{\text{TRUE}}}{\hat{m}}. \tag{4.11}$$

It is clear from Table 4.1 that using the screening step (NMCD+) significantly improves the computational cost of NMCD. However using this screening step comes at a cost of not correctly detecting the true changepoints. It can be seen that in all cases NMCD+ detects fewer true positives and more false positives than NMCD.

	True Discovery Rate			False Discovery Rate			Time		
(I)	NMCD	NMCD+	ED-PELT	NMCD	NMCD+	ED-PELT	NMCD (min)	NMCD+ (s)	ED-PELT (s)
$N(0,1)$	0.927 (0.002)	0.912 (0.003)	0.924 (0.002)	0.073 (0.002)	0.088 (0.003)	0.076 (0.002)	19.403 (0.051)	1.677 (0.007)	0.102 (0.003)
$t_{(3)}$	0.803 (0.004)	0.763 (0.004)	0.796 (0.004)	0.233 (0.004)	0.240 (0.004)	0.210 (0.004)	21.037 (0.065)	1.685 (0.008)	0.129 (0.003)
$\chi^2_{(3)}$	0.908 (0.003)	0.834 (0.003)	0.911 (0.003)	0.097 (0.003)	0.166 (0.003)	0.091 (0.003)	19.623 (0.040)	1.651 (0.006)	0.159 (0.001)
(II)									
$N(0,1)$	0.580 (0.007)	0.398 (0.005)	0.583 (0.007)	0.454 (0.007)	0.609 (0.005)	0.424 (0.007)	19.963 (0.061)	1.577 (0.002)	0.288 (0.006)
$t_{(3)}$	0.482 (0.006)	0.323 (0.005)	0.487 (0.006)	0.567 (0.006)	0.695 (0.005)	0.527 (0.006)	10.732 (0.088)	1.578 (0.001)	0.216 (0.002)
$\chi^2_{(3)}$	0.492 (0.006)	0.390 (0.005)	0.502 (0.006)	0.513 (0.007)	0.612 (0.005)	0.498 (0.006)	22.113 (0.050)	1.638 (0.002)	0.317 (0.002)
(III)	0.477 (0.008)	0.363 (0.008)	0.477 (0.002)	0.531 (0.008)	0.640 (0.008)	0.524 (0.002)	24.717 (0.014)	1.516 (0.015)	0.351 (0.002)

Table 4.1: True and false discovery rates and time comparisons for NCMD, NMCD+ and ED-PELT. Values in the table are mean (standard errors in parentheses) for 100 replications.

These measures provide a good evaluation of the number as well as location of changepoints. In order to explore the accuracy of the changepoint locations further we can use the distance measures as in Zou et al. (2014). That is we can use the worst case distance between the true changepoint set and the false changepoint set as in Boysen et al. (2009). If we set $\boldsymbol{\tau}$ as the set of true changepoints and $\hat{\boldsymbol{\tau}}$ as the set of detected changepoints then the over-segmentation and under-segmentation errors are calculated, respectively, as:

$$d(\hat{\boldsymbol{\tau}}, \boldsymbol{\tau}) = \max_{1 \leq i \leq \hat{m}} \min_{1 \leq j \leq m} |\hat{\tau}_i - \tau_j| \quad \text{and} \quad d(\boldsymbol{\tau}, \hat{\boldsymbol{\tau}}) = \max_{1 \leq j \leq m} \min_{1 \leq i \leq \hat{m}} |\tau_j - \hat{\tau}_i|. \tag{4.12}$$

Table 4.2 gives the average of over-segmentation and under-segmentation errors for NMCD and NMCD+ as well as the number of detected changepoints. The over segmentation error is higher for NMCD+ than it is for NMCD in all models. In model 1 with the normal errors both NMCD and NMCD+ found the same number of changepoints for all models but on average NMCD was more accurate than NMCD+. The under segmentation error is comparable for both NMCD and NMCD+ for all models except model 1 with Student's- t distribution error where the under segmentation error is much higher for NMCD than NMCD+ and in model 2 with chi-squared error where the under-segmentation error is much higher for NMCD+ than NMCD. In all cases NMCD+ found the same or less number of changepoints than NMCD but closer to the true number. However even though NMCD+ detected the true number of changepoints more we see that the locations of these changepoints were most of the time less accurate than NMCD.

	Over Segmentation error			Under Segmentation error			Number of detected changepoints		
(I)	NMCD	NMCD+	ED-PELT	NMCD	NMCD+	ED-PELT	NMCD	NMCD+	ED-PELT
$N(0,1)$	1.080 (0.055)	1.170 (0.042)	1.280 (0.063)	1.080 (0.055)	1.170 (0.042)	1.280 (0.063)	11.000 (0.000)	11.000 (0.000)	11.000 (0.000)
$t_{(3)}$	2.850 (0.096)	3.120 (0.097)	2.530 (0.077)	14.420 (0.822)	5.000 (0.382)	2.860 (0.093)	11.560 (0.091)	11.040 (0.020)	11.100 (0.036)
$\chi^2_{(3)}$	0.970 (0.029)	2.290 (0.059)	0.990 (0.032)	2.490 (0.343)	2.290 (0.059)	1.030 (0.033)	11.070 (0.033)	11.000 (0.000)	11.030 (0.017)
(II)									
$N(0,1)$	5.160 (0.194)	8.660 (0.217)	4.860 (0.175)	13.680 (0.768)	12.080 (0.494)	5.780 (0.269)	4.290 (0.057)	4.090 (0.032)	4.060 (0.028)
$t_{(3)}$	9.940 (0.286)	12.760 (0.267)	10.980 (0.354)	23.830 (0.896)	23.450 (0.764)	16.280 (0.622)	4.590 (0.091)	4.300 (0.063)	4.160 (0.047)
$\chi^2_{(3)}$	6.800 (0.242)	9.630 (0.247)	7.090 (0.238)	7.730 (0.317)	10.310 (0.320)	7.090 (0.238)	4.070 (0.029)	4.010 (0.010)	4.000 (0.000)
(III)									
	2.730 (0.076)	5.730 (0.152)	3.030 (0.104)	4.730 (0.386)	6.000 (0.165)	3.240 (0.121)	3.060 (0.028)	3.020 (0.014)	3.010 (0.010)

Table 4.2: Over-segmentation and under-segmentation errors, and the number of changepoints detected for NCMD, NMCD+ and ED-PELT. Values in the table are mean (standard errors in parentheses) for 100 replications.

4.4.2 Size of Screening Window

We now turn to consider the choice for the size of the screening window N_I further. Using model 1 with normal errors we can compare the results for different values of N_I. The default value for this data is $N_I = 10$, but we now repeat the analysis using $N_I \in \{1, \ldots, 20\}$. Figure 4.1a shows a bar plot of the number of times (in 100 simulations) that the window size resulted in the same changepoints as using NMCD without screening. Figure 4.1b shows the number of changepoints detected with different window lengths and Figure 4.1c looks at the number of true and false positives found using the different window lengths in the screening step. Figure 4.1d shows the computational time taken for NMCD+ with varying window lengths N_I. We found similar results for the other models.

It is clear that whilst in the majority of the cases NMCD+ with the different N_I finds the correct number of changepoints the location of these are not always correct and in fact are different than that found using NMCD. It is also worth noting that even though many window sizes find 11 changepoints the location of these may be different for different window lengths. In general the performance decreases as window length increases however the results do fluctuate a bit. This shows that the performance of NMCD+ is sensitive to the choice of the window size. Despite this we can see that NMCD+ is significantly faster than NMCD especially as the window length increases.

The NMCD method also requires us to choose a penalty value in order to pick the best segmentation. The default choice appears to work reasonably well, but resulted in slight over-estimates of the number of changepoints for our three simulation scenarios. These over-estimates suggest that the penalty value has been too small.

4.4.3 Choice of K in ED-PELT

We now turn our attention to ED-PELT. In order to use the improvement suggested in Section 4.3.1 for ED-PELT we first of all need to decide on an appropriate value for K. We use model 1 again to assess the performance of ED-PELT using only K

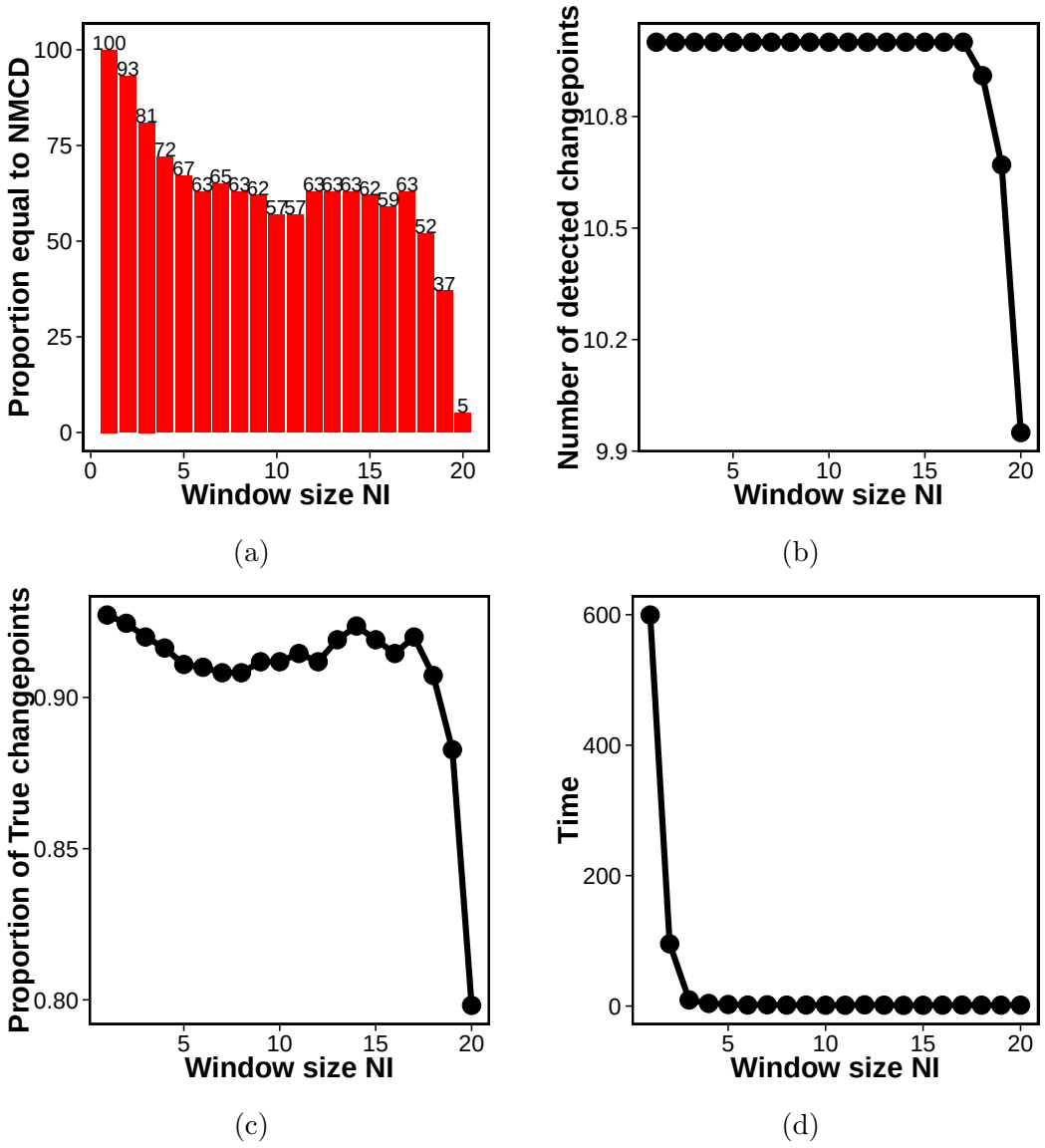
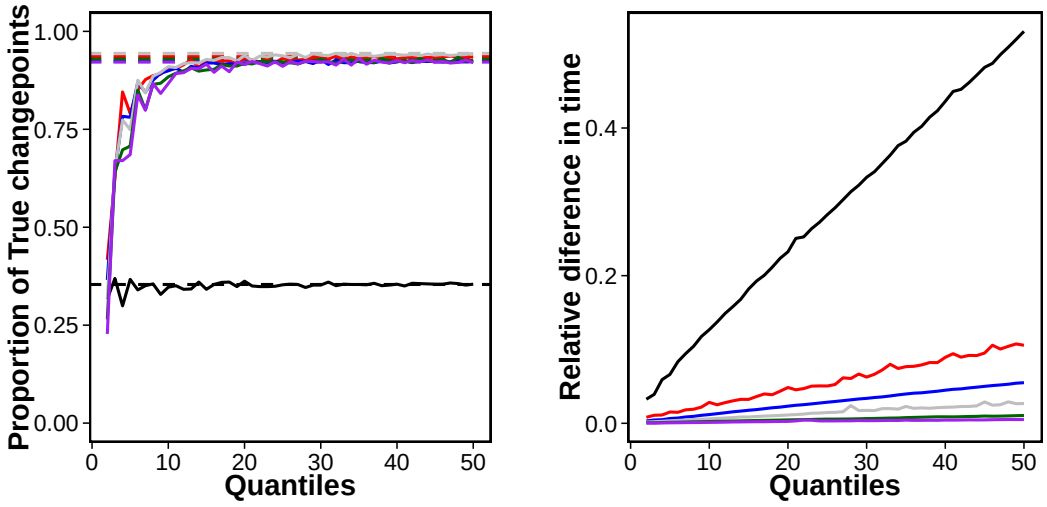


Figure 4.1: (a) The number of replications out of 100 in which using NMCD+ with varying N_I results in the same results as NMCD without screening. (b) The number of changepoints detected with with increasing window size N_I. (c) The proportion of true changepoints detected with varying window size N_I. (d) The computational time (secs) for NMCD+ with increasing window size N_I.



(a) (b)

Figure 4.2: (a) The proportion of true positive changepoints for a range of quantiles, K, in ED-PELT (solid) in comparison to ED-PELT (dashed). Black: $n = 100$, red: $n = 500$, blue: $n = 1000$, grey: $n = 2000$ and dark green: $n = 5000$. (b) Relative speed of using ED-PELT compared to using ED-PELT. with varying number of quantiles, K. Black: $n = 100$, red: $n = 500$, blue: $n = 1000$, grey: $n = 3000$, dark green $n = 5000$ and purple: $n = 10000$.

quantiles of the data, for a range of values of K, in comparison to ED-PELT using the full data-set. Here we only look at the model with normal errors and simulate data-series with lengths $n = (100, 500, 1000, 2000, 5000, 10000)$. Further simulations using different error terms gave similar results. In order to assess performance we look at the proportion of true positives detected using both methods and also the computational cost. Again we use 100 replications. The results for the accuracy can be seen in Figure 4.2a.

We can see from Figure 4.2a that as the number of quantiles increases the proportion of true change points detected using the approximated ED-PELT converges to the same result as the full version of ED-PELT . As the length of the data increases this convergence appears to happen more slowly, this can be seen from the purple line in Figure 4.2a, which represents data of length 10000. We suggest using $K = \lceil 4 \log(n) \rceil$ in order to conserve as much accuracy as possible. This choice corresponds to $K = 19, 25, 28, 31, 35$ and 37 for $n = 100, 500, 1000, 2000, 5000$ and 10000

respectively.

In addition to the accuracy we also look at the relative speed up of ED-PELT with various K values in comparison to ED-PELT, i.e.,

$$\frac{\text{(speed of approximated ED-PELT)}}{\text{speed of full ED-PELT}}.$$

The results of this analysis can be seen in Figure 4.2b. Clearly as the number of quantiles increases the relative speed up decreases. This is expected since the number of quantiles is converging to the whole data-set which is used in ED-PELT. We can also see that the relative speed up of ED-PELT increases with increasing data length.

4.4.4 Comparison of NMCD and ED-PELT

We next compare ED-PELT with $K = 4\log(n)$ to NMCD as above. For this we perform an equivalent analysis to that of Section 4.4.1 and again look at the accuracy of the methods and the computational time. As before, to implement NMCD we used the `nmcdr` R package (Zou and Zhange, 2014) which is written in FORTRAN with an R user interface. We use the `changepoint.np` R package (Haynes, 2016) to run ED-PELT which also has an R interface but with the main body of the code written in C. We use the default parameters for nmcd and for ED-PELT we use the SIC/BIC penalty term, $2p\log(n)$, where p is the number of parameters, to match the penalty term used in the NMCD algorithm.

The results for ED-PELT can be found in Tables 4.1 and 4.2. In terms of accuracy we can see that ED-PELT is comparable to NMCD albeit lacking slightly in some of the measures, however it is significantly faster to run. We can also see from table 4.2 that ED-PELT has lower under-segmentation error than NMCD in most of the models, however it has a higher segmentation error. In comparison to NMCD+, ED-PELT is faster and also more accurate so would be the better approximate method to choose.

4.5 Activity Tracking

In this section we apply ED-PELT to try to detect changes in heart-rate during a run. Wearable activity trackers are becoming increasingly popular devices used to record step count, distances (based on the step count), sleep patterns and in some of the newer devices, such as the Fitbit change HR (Fitbit Inc., San Francisco, CA), heart-rate. The idea behind these devices is that the ability to monitor your activity should help you lead a fit and active lifestyle. Changepoint detection can be used in daily activity tracking data to segment the day into periods of activity, rest and sleep.

Similarly, many keen athletes, both professional and amateur, also use GPS sports watches which have the additional features of recording distance and speed which can be very beneficial in training, especially in sports such as running and cycling. Heart-rate monitoring during training can help make sure you are training hard enough without over training and burning out. Heart-rate is the number of heart beats per unit time, normally we express this as beats per minute (bpm).

4.5.1 Changepoints in Heart-Rate Data

In the changepoint and signal processing literature many authors have looked at heart-rate monitoring in different scenarios (see for example Khalfa et al. (2012); Galway et al. (2011); Billat et al. (2009); Staudacher et al. (2005)). Aubert et al. (2003) give a detailed review of the influence of heart-rate variability in athletes. They highlight the difficulty of analysing heart-rate measurements during exercise since no steady state is obtained due to the heart-rate variability increasing according to the intensity of the exercise. They note that one possible solution is to pre-process the data to remove the trend.

In this section we apply ED-PELT to see whether changes can be detected in the raw heart-rate time-series without having to initially pre-process the data. We use a nonparametric approach since heart-rate is a stochastic time dependent series

and thus does not satisfy the conditions for an IID normal model. However we will compare the performance had we assumed that the data was normal in Section 4.5.3. The aim is to develop a method which can be used on data recorded from commercially available devices without the need to pre-process the data.

4.5.2 Range of Penalties

One disadvantage of ED-PELT over NMCD is that ED-PELT produces a single segmentation, which is optimal for the pre-chosen penalty value ξ_n. By comparison, NCMD finds a range of segmentations, one for each of $m = 1, \ldots, M$ changepoints (though, in practice, the `nmcdr` package only outputs a single segmentation). Whilst there are default choices for ξ_n, these do not always work well especially in real-life applications where the assumptions they are based on do not hold. There are also advantages to being able to compare segmentations with different number of changepoints.

Haynes et al. (2017) propose a method, Changepoints over a Range Of PenaltieS (CROPS), which efficiently finds all the optimal segmentations for penalty values across a continuous range. This involves an iterative procedure which chooses values of ξ_n to run PELT on, based on the segmentations obtained from previous runs of PELT for different penalty values. Assume we have a given range $[\xi_{\min}, \xi_{\max}]$ for the penalty value, and the optimal segmentations at $\xi_{\min}$ and $\xi_{\max}$ have $m_{\min}$ and $m_{\max}$ changepoints respectively. Then CROPS requires at most $m_{\min} - m_{\max} + 2$ runs of PELT to be guaranteed to find all optimal segmentations for $\xi_n \in [\xi_{\min}, \xi_{\max}]$. Furthermore, it is possible to recycle many of the calculations from early runs of PELT to speed up the later runs.

Nonparametric Changepoint Detection

An example data-set is given in Figure 4.4, where we show heart-rate, speed and elevation recorded during a 10 mile run. We will aim to segment this data using the

heart-rate data only, but include the other two series in order that we may assess how well the segmentation of the heart-rate data relates to the obvious different phases of the run. As is common in nonparametric methods, ED-PELT assumes that data is IID which in the case of heart-rate data the assumptions do not hold since there is some time-series dependence between segments. However for the moment we will assume all the assumptions hold and that we can use this method. In training many people use heart-rate as an indicator of how hard they are working. There are different heart-rate zones that you can train in each of which enhances different aspects of your fitness (BrainMacSportsCoach, 2015). The training zones are defined in terms of percentages of a maximum heart-rate: peak (90-100%), anaerobic (80-90%), aerobic (70-80%) and recovery ($< 70\%$).

This example looks at detecting changes in heart-rate over a long undulating run. We use CROPS with ED-PELT with $\xi_{min} = 25$, $\xi_{max} = 200$ and $K = 4\log(n)$ (the results are similar for different K). In order to choose the best segmentation we use the approach suggested by Lavielle (2005). This involves plotting the segmentation cost against the number of changepoints and then looking for an "elbow" in the plot. The points on the "elbow" are then suggested to be the most feasible segmentations. The intuition for this method is that as more true changepoints are detected the cost will decrease however as we detect more changepoints we are likely to be detecting false positives and as such the cost will not decrease as much. The plot of the "elbow" for this example can be seen in Figure 4.3a. The elbow is not always obvious therefore the choice can be subjective, in high throughput situations you can often learn a good choice of penalty through comparing segmentations for a range of training data-sets (see Hocking et al. (2013a)). However in this example the elbow approach gives us a method for roughly choosing the best segmentations which we can then explore further. We have highlighted the points on the "elbow" as the points which are between the two red lines.

We decided from this plot that the segmentations with 9, 10, 12 and 13 change-

(a) Elbow plot for ED-PELT (b) Elbow plot for a change in slope

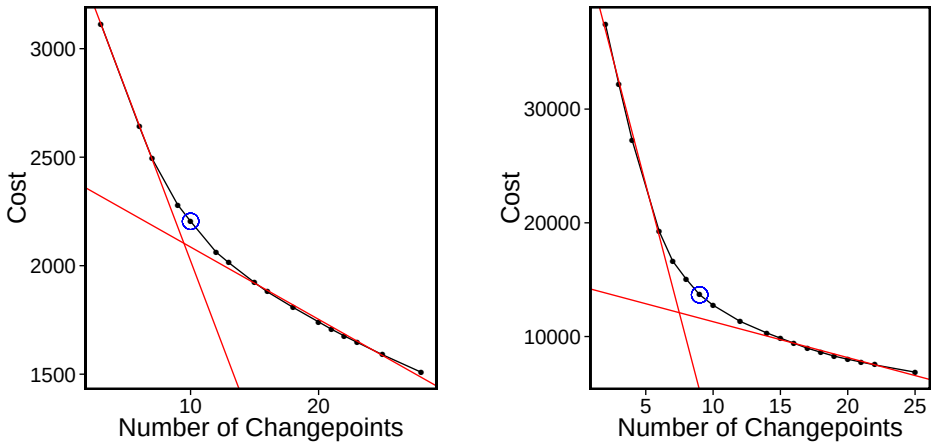


Figure 4.3: The cost vs number of changepoints plotted for (a) ED-PELT and (b) Change in slope. the red lines indicate the elbow and the blue circle highlights the point that we use as being the centre of the elbow.

points are the best. We illustrate the segmentation with 10 changepoints, the number of changepoints at the centre of the elbow in Figure 4.3a indicated by the blue circle, in Figure 4.4. The segments have been colour coded based on the average heart-rate in each segment. That is red: peak, orange: anaerobic, yellow: aerobic and green: recovery. Alternative segmentations from the number of changepoints on the elbow can be found in Appendix C.

We superimpose the changepoints detected in the heart-rate onto the plots for speed and elevation to see if we can explain any of the changepoints. The first segment captures the "warm-up" where the heart-rate is on average in the recovery zone but is rising to the anaerobic zone. The heart-rate in the second segment is in the anaerobic zone but changes to the peak zone in segment three. This change initially corresponds to an increase in speed and then it is because of the steep incline. The third changepoint matches up to the top of the elevation which is the start of the fourth segment where the heart-rate drops into the anaerobic zone whilst running downhill. The fifth segment is red which might be as a result of both the speed being slightly higher than the previous segment and consistent, and a slight incline in ele-

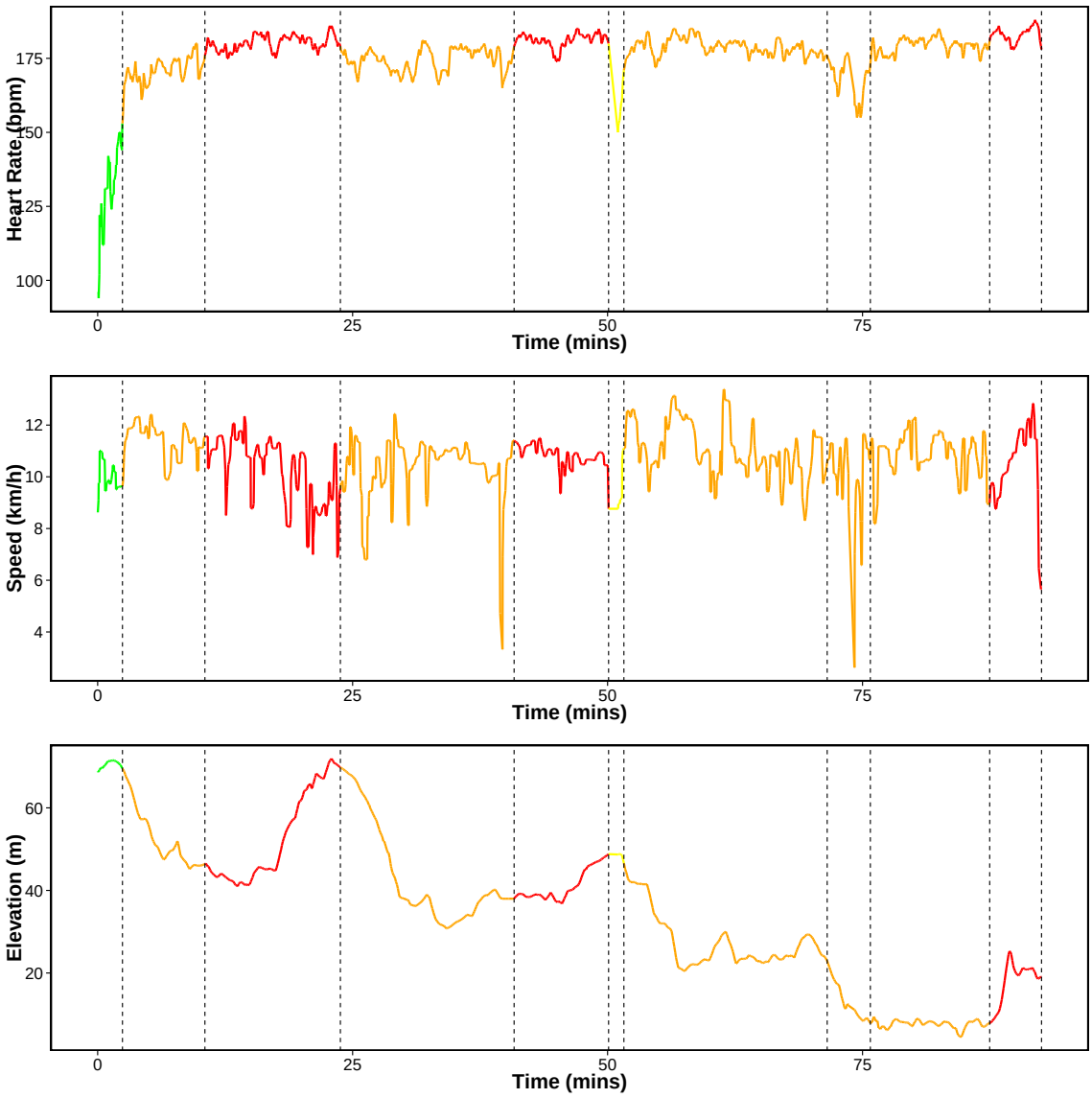


Figure 4.4: Segmentations using ED-PELT with 10 changepoints. We have colour coded the line based on the average heart-rate of each segment where red: peak, orange: anaerobic, yellow: aerobic and green: recovery.

vation. This is followed by a brief time in the aerobic zone which could be due to a drop in speed. The heart-rate in the next three segments stays in the anaerobic zone. The changepoints that split this section into three segments relate to the dip in speed around 75 minutes. In the final segment the heart-rate is in the peak zone which corresponds to an increase in elevation and an increase in speed (a sprint finish). We believe ED-PELT has found sensible segmentations that can be related to different phases of the run and regimes in heart-rate activity despite the data not satisfying the independence assumption.

4.5.3 Piece-wise Linear Model

For comparison we look at estimating the changepoints based on a penalised likelihood approach that assumes the data is normally distributed with a mean that is piecewise linear within each segment. To find the best segmentation we use PELT with a segment cost proportional to minus the log-likelihood of our model:

$$\mathcal{C}(y_{s:t}) = \min_{\theta_1,\theta_2} \sum_{u=s}^{t} (y_u - \theta_1 - u\theta_2)^2, \tag{4.13}$$

where θ_1 and θ_2 and the estimates of the segment intercept and slope, respectively. We use CROPS to find the best segmentation under this criteria for a range of penalties. The resulting elbow plot can be seen in Figure 4.3b. We can see that the number of changepoints for the feasible segmentations is similar to the number of changepoints using ED-PELT. Figure 4.5 shows the segmentation with 9 changepoints which we have deduced to being the number of changepoints in the centre of the elbow in Figure 4.3b. Alternative segmentations from the number of changepoints on the elbow can be found in Appendix C.

It is obvious from the first look at Figure 4.5 that the change in slope method has not detected segments where the average heart-rate is different to the surrounding segments. The majority of the plot is coloured orange with only changes in the

first and last segments. The change in slope method splits the "warm-up" period into two segments whereas having this as one segment appears more appropriate. Unlike ED-PELT the change in slope does not detect changes which correspond to the change in elevation and thus ED-PELT appears to split the heart-rate data into more appropriate segments which relate to different phases of the run.

4.6 Conclusion

We have developed a new algorithm, ED-PELT, to detect changes in data-series where we do not know the underlying distribution. Our method is an adaption of the NMCD method proposed by Zou et al. (2014) which defines the segment costs of a data-series based on the cumulative empirical distribution function and then uses an exact search algorithm to detect changes. The main advantage of ED-PELT over NMCD is that it is orders of magnitude faster. We initially reduced the time to calculate the cost of a segment from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$ by simplifying the definition of the segment cost by discrete approximation. To improve the computational cost Zou et al. (2014) use a screening step but as we show in Section 4.4 this is still slower than ED-PELT and less accurate. The main reason for this is we use an exact search algorithm, PELT (Killick et al., 2012), that uses inequality based pruning to reduce the number of calculations. This search algorithm is much quicker than the one used in Zou et al. (2014).

The main problem with PELT is it requires a penalty value to avoid under/over-fitting and the performance is detrimental to this choice. We overcome this problem by using CROPS (Haynes et al., 2017), which detects the changepoints for multiple penalty values over a continuous range. Future research could look at an alternative pruning method, cp3o, proposed by James and Matteson (2015) which used probabilistic pruning. This method does not require a penalty value however there are some mild conditions required for this search method which would need to be checked with the empirical distribution cost function.

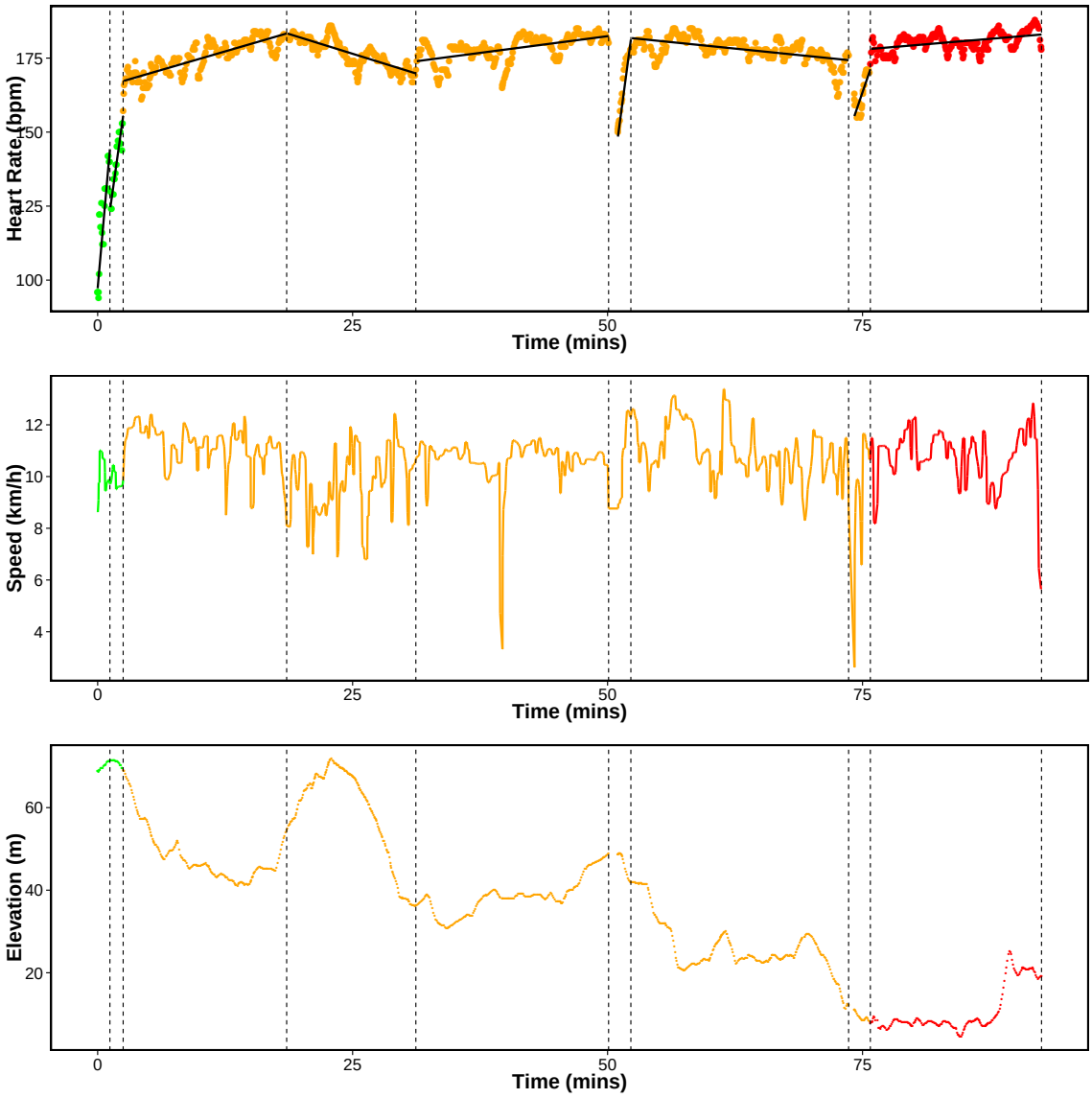


Figure 4.5: Segmentations using change in slope with 9 changepoints. We have colour coded the line based on the average heart-rate of each segment where red: peak, orange: anaerobic, yellow: aerobic and green: recovery. The solid black line in the top plot is the best fit for the mean within each segment.

We have also shown that nonparametric changepoint detection, using ED-PELT, holds promise for segmenting data from activity trackers. We applied our method to heart-rate data recorded during a run. As is common for current nonparametric approaches to changepoint detection, our method is based on the assumption that the data is independent and identically distributed within a segment. Despite this we were able to segment the data into meaningful segments, using an appropriately chosen penalty value, that correspond to different phases of the run and can be related to different regimes in heart-rate activity.

Code implementing ED-PELT is contained within the R library `changepoint.np` which is available on CRAN (Haynes, 2016).

Chapter 5

Parallel Changepoint Detection

5.1 Introduction

Over the past 20 or so years there have been phenomenal advances in technology and one result of this is the volume of data collected and stored has rapidly increased. High frequency data sensors are now common place, not just in specialised applications, but in the technology we use in our day to day lives, such as mobile phones.

There is a requirement to develop statistical methods that can cope with the sheer size of the data-sets available for analysis. In this chapter we think about applying changepoint detection in the realm of Big Data. There are many applications where detecting changepoints can be useful such as climatology (Reeves et al., 2007), neuroscience (Aston and Kirch, 2012) and linguistics (Kulkarni et al., 2015), to name a few. In terms of "Big Data", data-series of this volume arise in applications such as genomics, where the signal length of DNA copy number is typically of order $10^5 - 10^6$ (Rigaill, 2015), and in finance where information on different markets and locations are stored in a much finer scale, such as individual bids (Fan and Wang, 2007).

With advances in technology it is easier to get access to multi-processor systems whereby parallel computing can be used to share the analysis over multiple processors which will reduce the computational burden. There is very little work on parallel

changepoint detection and most of what there is is concerned with parallelising over dimensions in multivariate changepoint detection. This is a completely different set-up to the one we are interested in, in this chapter. The Binary Segmentation type approaches, introduced in Section 2.4 are "embarrassingly parallel" since it is obvious how they can be run in parallel. At each stage of the iteration a single changepoint is searched for over subsets of data independent of one another. This process can be done simultaneously over the multiple processors.

Nikol'skii and Furmanov (2016) propose an approximate algorithm to detect changes in parallel. This method first of all splits the data into equal sized segments and uses a t-test statistic to test the hypothesis of equal means across all the random variables in each segment. If the hypothesis is rejected then the method checks for a changepoint using the Schwartz Information Criteria (Schwarz, 1978). That is, a time-point, τ, is a changepoint if

$$\mathcal{C}(y_{1:\tau}) + \mathcal{C}(y_{\tau+1:n}) + \log(n) < \mathcal{C}(y_{1:n}), \tag{5.1}$$

where $\mathcal{C}$ is some cost such as twice the negative log-likelihood. If no changepoints are found in adjacent segments then the algorithm uses points around the boundaries of the segments to check for the existence of changepoints. The huge flaw in this method is the assumption that only one changepoint exists on the data sent to each core and thus makes this approach an infeasible changepoint detection method as there is absolutely no guarantee of detecting all of the changes.

It is not as simple to parallelise the dynamic programming methods, discussed in Section 2.5.2 since each stage of the algorithm requires calculations from previous iterations. This is the problem we are interested in. Our main contribution in this chapter is the proposal of two methods for parallel changepoint detection which use a "split and merge" approach. Split and merge approaches basically split the data evenly across the multiple processors, run some analysis and then somehow merge the data to give a full solution. This approach has been used across different areas of

statistics with the main challenge being merging the data in the final step. Matloff (2016); Hegland et al. (1999) and Fan et al. (2007) use an average, or weighted average, approach to merge estimators of IID random variables. Under the assumption the data is IID, Matloff (2016) show that this method is asymptotically consistent for statistical methods such as quantile regression and hazard function estimation.

The averaging approach considered by Matloff (2016) will not work in all examples and in particular it will not make sense in changepoint detection to average the detected changepoints from the different subsets. Song and Liang (2015) propose an approach for Bayesian variable selection in which instead of averaging they perform Bayesian variable selection again on the aggregated variables from all of the subsets.

Throughout this chapter we will use the notation described in Section 2.2. The rest of this chapter is organised as follows. In Section 5.1.1 we will introduce the parallel set-up. In Section 5.2 we will discuss the Binary Segmentation type methods and show how they are embarrassingly parallel. This section will include a small simulation study to show the marginal time improvements, in one scenario, when the methods are parallelised. In Section 5.3 we introduce the dynamic programming algorithms and propose our two split and merge approaches. This section includes brief discussion on the consistencies of the estimators as well as highlighting a potential problem depending how the data is split across the cores. In Section 5.4 we do a comprehensive investigation into the performance of the two methods across many different scenarios.

5.1.1 Parallel Implementation

The simulations in this chapter were run on Intel processors with 70 cores. We have used the R programming language and the `doParallel` package (Calaway et al., 2014). For a review of some of the packages developed in R to provide communication to the various parallel infrastructures see Schmidberger et al. (2009) and for an up to date list of all the parallel packages in R see Eddelbuettel (2016). The `doParallel` package

provides a parallel back-end for the `foreach` package (Calaway et al., 2015) using the `parallel` package, which is inside R-core. The `foreach` package allows general iterations over elements without using a loop counter and thus allows for the loop to be easily implemented in parallel. The `foreach` package needs to be run in conjunction with a package such as `doParallel` in order to execute the code in parallel. The `parallel` package started as a merger of the `multicore` and `snow` (Tierney et al., 2015) packages, however much of the functionality of `multicore` has been integrated into `parallel`. The `multicore` package runs tasks on a single computer but it cannot be used on Windows as it requires an operating system that supports the fork system call. The `snow` package allows code to be run on a cluster.

The algorithms we propose in this chapter will be able to run on any parallel architecture, however it will require the user to modify the communication parts of the code to be specific to their parallel set-up. To register the `doParallel` package and start a cluster the following code (or equivalent) is required.

```
> cl <- makecluster(count=4)
> registerDoParallel(cl)
```

The part of the code to be run in parallel can be run using `foreach` using the %`dopar`% command

```
> (foreach(i = 1:(ncores)) %dopar% function(i),
```

where `i` can be used to give different information to the core such as a different part of the data-series to be analysed. We stop the cluster by using

```
> stopCluster(cl).
```

5.2 Embarrassingly Parallel

Binary Segmentation (Scott and Knott, 1974; Vostrikova, 1981) and its variants: Circular Binary Segmentation, CBS, (Olshen et al., 2004) and Wild Binary Segmentation,

WBS, (Fryzlewicz, 2014) fall into the category of embarrassingly parallel methods. That is it is obvious how to split the algorithm into smaller independent subtasks.

5.2.1 Binary Segmentation

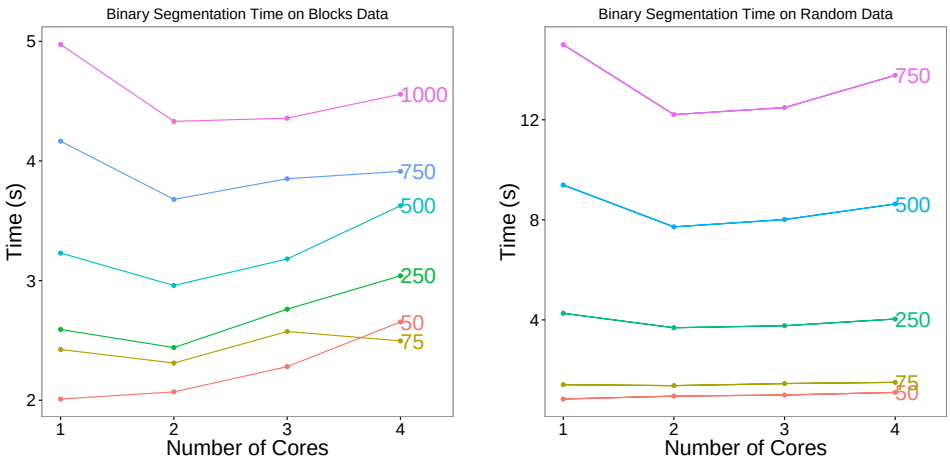
In BS we initially search the whole dataset for one changepoint, i.e., the point, τ, that satisfies the condition in (5.2) and also minimises the left hand side of (5.2),

$$\mathcal{C}(y_{1:\tau}) + \mathcal{C}(y_{\tau+1:n}) + \beta < \mathcal{C}(y_{1:n}), \tag{5.2}$$

where $\mathcal{C}(y_{s:t})$ is the cost from the data $y_s, ..., y_t$. The data is then split at τ and we search for a changepoint in the two new segments independently. This continues until no more changepoints are detected. Traditionally the calculations for the changepoints are computed in a loop over all of the segments on one processor. The computational cost may be improved by sending these calculations to multiple cores.

In theory this should speed up the calculations however BS is $\mathcal{O}(n \log n)$ so the overhead of scheduling the tasks and returning the results may mean it is not worth implementing in parallel. Additionally, the speed up will be more noticeable in situations where there are a large number of changepoints since at later stages of the algorithm more segments will be searched over for a change, so having multiple processors may be beneficial.

We explore the performance of parallelising BS in a couple of examples: one in which the number of changes is constant with increasing data length and one where the number of changes increases with increasing data length. In the first example we simulate the data from a blocks-signal (Donoho and Johnstone, 1994), with $m = 11$ changepoints for all data lengths. The signal has some Gaussian noise with variance equal to 1. We replicate this 100 times and the average time to run BS with different number of cores is shown in Figure 5.1a. The main thing to note from this is the time



(a) (b)

Figure 5.1: Computational time taken to run Binary Segmentation in parallel over multiple cores. (a) Is the blocks-signal with 11 changepoints for all data lengths. (b) Is the random data with increasing changepoints as the data length increases.

slowly increases with increasing number of cores. In this example there is not any benefit of parallelising.

In the second example we simulate data-sets with the number of changes increasing with increasing data length; for data-sets of length n there will be $m = n/100$ changes. We simulate the changepoints uniformly over time with the constraint that there must be atleast 20 time-points apart. To simulate the data we generate the segment means from a Gaussian distribution with mean 0 and standard deviation 5. We replicate this 100 times and the average time to run BS with different number of cores is shown in Figure 5.1b. This time there are marginal gains in speed if using 2 cores for larger data-sets. However these speed improvements are really small so it is probably not worth parallelising.

Variants of Binary Segmentation

Binary Segmentation lacks consistency due to its greedy nature. Fryzlewicz (2014) look at the asymptotic properties of BS with the cumulative sums (CUSUM) test statistic (Page, 1954) and show that as the number of data-points, $n \to \infty$, then BS is only asymptotically guaranteed to identify the true changepoints if the minimum

segment length is $\mathcal{O}(n^{3/4})$. Specifically changepoints are likely to be missed if they are close to another changepoint. Fryzlewicz (2014) propose a method, Wild Binary Segmentation (WBS) that aims to overcome the lack of consistency of BS. At each stage of BS, instead of calculating the global cost $\mathcal{C}(y_{1:n})$, WBS randomly draws a number of sub-samples, $y_{s:e}$, where $1 \leq s < e \leq n$, and detects a candidate changepoint within each sub-sample. The changepoint within each sub-sample that has the overall minimum cost is found to be the new changepoint and the data is now split here and the process is repeated, similar to BS.

The number of sub-samples chosen at each stage will affect the overall cost of this method. To improve the cost of WBS even further the computation of the single changepoints from the different sub-samples at each stage can be done over multiple cores. Thus WBS is trivially parallel and will be more amenable to parallelisation that BS since there are multiple calculations at every step of WBS that can be run simultaneously.

Another approach, Circular Binary Segmentation (CBS), was proposed by Olshen et al. (2004). This method uses an epidemic test statistic (Levin and Kline, 1985) to test for two changepoints in a segment instead of one as in the standard BS. The test statistic assumes that the mean before the first change and after the last change are the same. This is essentially the same as joining the endpoints of the segments, to make a circle, and then testing the mean of the arc between the changepoints against the mean of the compliment. To calculate the p-values in a non-normal setting they use a permutation approach to calculate reference distributions, however this is a computationally expensive approach which quadratically grows the with the number of changes.

Venkatraman and Olshen (2007) propose a couple of ways to speed up the computation of CBS however for additional speed up the permutation calculations at each stage of the algorithm can easily be parallelised.

5.3 Dynamic Programming Algorithms

Another class of changepoint detection algorithms involve using dynamic programming approaches (Bellman, 1957), as described in Section 2.5.2. These approaches have the advantage over the above Binary Segmentation type methods that they solve the minimisation problem exactly.

As a reminder, one of the dynamic programming approaches, Optimal Partitioning (Jackson et al., 2005), involves solving the following recursion:

$$F(t) = \min_{m,\tau_{1:m}} \left\{ \sum_{i=1}^{m+1} \mathcal{C}(y_{(\tau_{i-1}+1):\tau_i}) + \beta \right\} = \min_{s \in \{0,\ldots,t-1\}} \{F(s) + \mathcal{C}(y_{(s+1):t}) + \beta\}, \quad (5.3)$$

where $s < t$. That is, the minimum cost for the data $y_{1:t}$ is the minimum cost for segmenting the data up to the last changepoint s plus the cost of the segment $y_{(s+1):t}$. The value of s that attains the minimum of (5.3) is the position of the last changepoint before t. These recursions are solved for $t = 1, 2, ..., n$ with a computational cost $\mathcal{O}(n^2)$. The segmentations can be recovered by

$$\tau_t = \arg \min_{s \in \{0,\ldots,t-1\}} \{F(s) + \mathcal{C}(y_{(s+1):t}) + \beta\}, \quad (5.4)$$

which gives the optimal location of the last changepoint in $y_{1:t}$. Extracting the full set of changepoints in the optimal segmentation is then achieved by a simple recursion backwards through the data.

5.3.1 Parallelisation

Due to the recursive nature of the calculations, to solve the optimisation problem, we are unable to split the calculations up, as in Section 5.2, since at each point, y_t, we require the calculations from $y_{1:(t-1)}$. Instead, we propose two methods which split the data into smaller subsets, calculate the changepoints for these subsets, and then

merge the results to find a complete set of changepoints for the whole data-set.

For both methods, we first of all are required to calculate some summaries, $\mathcal{S}$, of the whole data-set. For the case of a change in mean this is just the cumulative sums of the data:

$$\begin{aligned} \mathcal{S}(0) &= 0, \\ \mathcal{S}(i) &= \mathcal{S}(i-1) + y_i, \end{aligned} \tag{5.5}$$

for $1 \leq i \leq n$.

5.3.2 Approach 1

The most obvious way to split the data is as follows:

Step 1 Given L CPUs split the data such that the kth CPU has the points $y_{\frac{n}{L}\times(k-1)+1:\frac{n}{L}\times k}$ and detect the changepoints in this subset of data. This can be done using the recursion in (5.3) where we use the summaries calculated over the whole data-set, $\mathcal{S}$, to calculate the segmentation costs $\mathcal{C}$. That is, in the case of a change in mean

$$\mathcal{C}(y_{s:t}) = \frac{\{\mathcal{S}(t) - \mathcal{S}(s)\}^2}{(t-s+1)}. \tag{5.6}$$

We define $\{\hat{\tau}_s^k\}_{s=1}^{\hat{m}_k} = \{\frac{n}{L} \times (k-1) = \hat{\tau}_0^k < \hat{\tau}_1^k < \hat{\tau}_2^k < ... < \hat{\tau}_{\hat{m}_k+1}^k = \frac{n}{L} \times k\}$ as the set of $\hat{m}_k$ changepoints detected on the kth CPU.

Step 2 From each of the L batches we have a set of possible changepoints. We can now fit a model but only allow changes at the location of changepoints detected in step 1, i.e., $y_{\{\hat{\tau}_s^1\}_{s=1}^{\hat{m}_1}}, y_{\{\hat{\tau}_s^2\}_{s=1}^{\hat{m}_2}}, ..., y_{\{\hat{\tau}_s^L\}_{s=1}^{\hat{m}_L}}$. Again we use the summaries calculated across the whole data-set to calculate the segment costs.

Solving the recursion in (5.3) for the data on each core will have a cost of $\mathcal{O}(\frac{n^2}{L^2})$. Step 2 will depend on how many changes are found in step 1. If m changes are found

then the cost for step 2 is $\mathcal{O}\{(m+L)^2\}$. The more cores we use then the lower the cost in step 1 however the higher the cost in step 2. The best number of cores to use will be the one that minimises the total cost. In terms of how we would want L to increase as n increases, the optimal rate will balance the CPU cost from the split and merge steps. This would have $L = \mathcal{O}(n^{1/2})$ and the resulting algorithm would then be $\mathcal{O}(n)$. This suggests that there will not be much gain in speed in situations where the algorithms are $\mathcal{O}(n)$. For example the PELT algorithm is $\mathcal{O}(n)$ when the number of changes is linear in n.

From herein we will refer to this method as SM1 (Split and Merge 1). These methods proposed are analogous to split and merge approaches used in other areas of statistics such as the one proposed by Song and Liang (2015) for Bayesian variable selection.

This approach may suffer from the same drawback of Nikol'skii and Furmanov (2016), that was discussed in the introduction. Namely it will struggle to detect changepoints that are near to the boundary of one of the subsets of data. We suggest an approach to overcome this drawback later in Section 5.3.4.

5.3.3 Approach 2

There is another way that we can split the data that avoids the boundary issue of SM1. That is:

Step 1 Given L CPUs, the kth CPU fits a model with changes allowed at points: $y_{k+L}, y_{k+2L}, ...$, etc, This can be done using the summaries calculated across the entire data-set, $\mathcal{S}$.

Step 2 From each of the L batches we have a set of the possible changepoints. We can now fit a model but only allow changes at this new subset of points. This is the same as in Step 2 of Approach 1.

Given m actual changepoints and n data-points the cost per CPU for solving the recursion in (5.3) is $\mathcal{O}(\frac{n^2}{L^2})$ and the cost for the final step is $\mathcal{O}\{(mL)^2\}$. From herein we will refer to this method as SM2 (Split and Merge 2).

5.3.4 Boundaries

The power of changepoint detection is proportional to some multiple of the size of the change and the segment length (Frick et al., 2014). There may be problems if the changes occur close to the splits in SM1. To increase the power in SM1 we can consider the points around the boundary of where 2 subsets of data meet in step 2. That is in step 2 we now allow the points

$y_{\{\hat{\tau}_s^1-B:\hat{\tau}_s^1:\hat{\tau}_s^1+B\}_{s=1}^{\hat{m}_1}} : y_{\{\hat{\tau}_s^2-B:\hat{\tau}_s^2:\hat{\tau}_s^2+B\}_{s=1}^{\hat{m}_2}}, ..., y_{\{\hat{\tau}_s^L-B:\hat{\tau}_s^L:\hat{\tau}_s^L+B\}_{s=1}^{\hat{m}_L}}$ to be candidate changepoint locations where B is the number of points to include. Obviously the greater the size of boundary the more chance we have of detecting the changepoints but having too many data-points will contradict the improved cost of parallelisation. We explore different choices for this boundary in Section 5.4.3.

5.4 Simulations

In this section we empirically investigate the performance of SM1 and SM2 under different scenarios.

5.4.1 Signals

The test signals which we will use are:

teeth : Data-sets which have a "teeth" pattern; equidistant changepoints and equal jump sizes, jumps alternate between decreasing and increasing.

stairs : Data-sets which have a "stairs" pattern; equidistant changepoints and equal jump sizes which are always increasing to a point and then they decrease.

blocks : We use the popular test signal as in Donoho and Johnstone (1994) For different segment lengths n we have changepoints at $(\{0.1, 0.13, 0.15, 0.23, 0.25, 0.40, 0.44, 0.65, 0.76, 0.78, 0.81\}) \times n$ and segment means $\{0, 4, -1, 2, -2, 3, -1.2, 0.9, 5.2, 2.1, 4.2, 0\}$.

mix : This signal has more prominent changepoints between smaller segment lengths and less prominent changepoints between larger segments, thus all changepoints can be detected consistently. We use this test signal to explore the time of the methods. For segment length n it has changepoints at $(\{11, 21, 41, 61, 91, 121, 161, 201, 251, 301, 361, 421, 491, 561, 631, 701, 761, 821, 871, 921, 961, 1001, 1031, 1061, 1081, 1101, 1111, 1121\}/1121) \times n$ and segment means $\{7, -7, 6, -6, 5, -5, 4, -4, 3, -3, 2, -2, 1, -1, 1, -2, 2, -3, 3, -4, 4, -5, 5, -6, 6, -7, 7\}$.

Random : We generate data-sets where the changepoints and parameters are randomly generated. We simulate these data-sets such that the detection difficulty of each jump is the same (Pein et al., 2015) and the changepoints are not too close (Killick et al., 2012). To do this we:

1. Fix the length of the data, n, the number of changepoints, K, and a constant, C. We also need to decide on the minimum length a segment can be, $minseglen$.
2. Draw the locations of the changepoints such that they are uniformly distributed with the restriction $\min_{0,\ldots,K} |\tau_{k+1} - \tau_k| \geq minseglen$.
3. We then let the standard deviation on the segment, σ_i, be 2^{U_i} where U_i is uniformly distributed on [-2,2].
4. We can then determine the values of the mean on the segments, μ_i. To do

this we let $\mu_0 = 0$ and then:

$$|\mu_i - \mu_{i-1}| = \sqrt{\frac{C}{n} \min\left(\frac{\tau_{i+1} - \tau_i}{\sigma_i^2}, \frac{\tau_i - \tau_{i-1}}{\sigma_{i-1}^2}\right)^{-1}}, \tag{5.7}$$

where C is a constant which controls the difficulty of the change. We choose randomly with probability $1/2$ whether the mean increases or decreases at the change.

Examples of these signals can be seen in Figure 5.2.

5.4.2 Evaluation

To evaluate the detected changes we use the *True and False Discovery Rate (TFDR)* which looks at the number and location of the changepoints. We define a detected changepoint, $\hat{\tau}_i$, as true if for some threshold value, h,

$$\min_{i \leq j \leq m} \{|\hat{\tau}_i - \tau_j|\} \leq h. \tag{5.8}$$

In this case we set the threshold value to 0. For the results we want to compare the detected changepoints using the split and merge approaches to the changes detected using changepoint detection on one core. Thus in this case the set of "true" changepoints, $\{\tau_j\}_{j=1}^m$, are the changes detected on one core. Since we are dealing with exact algorithms the changepoints can be detected using any of the algorithms such as Optimal Partitioning, PELT or FPOP as they will all find the same set of changes.

In the simulations we use PELT since FPOP has been shown to have an empirical cost which is comparable with Binary Segmentation and thus the speed gains, if any, will be marginal like what we saw in Section 5.2. PELT, however, can be applied to a greater range of problems, specifically those which have a change in more than one variable and hence we are interested in speeding up this algorithm. PELT has a

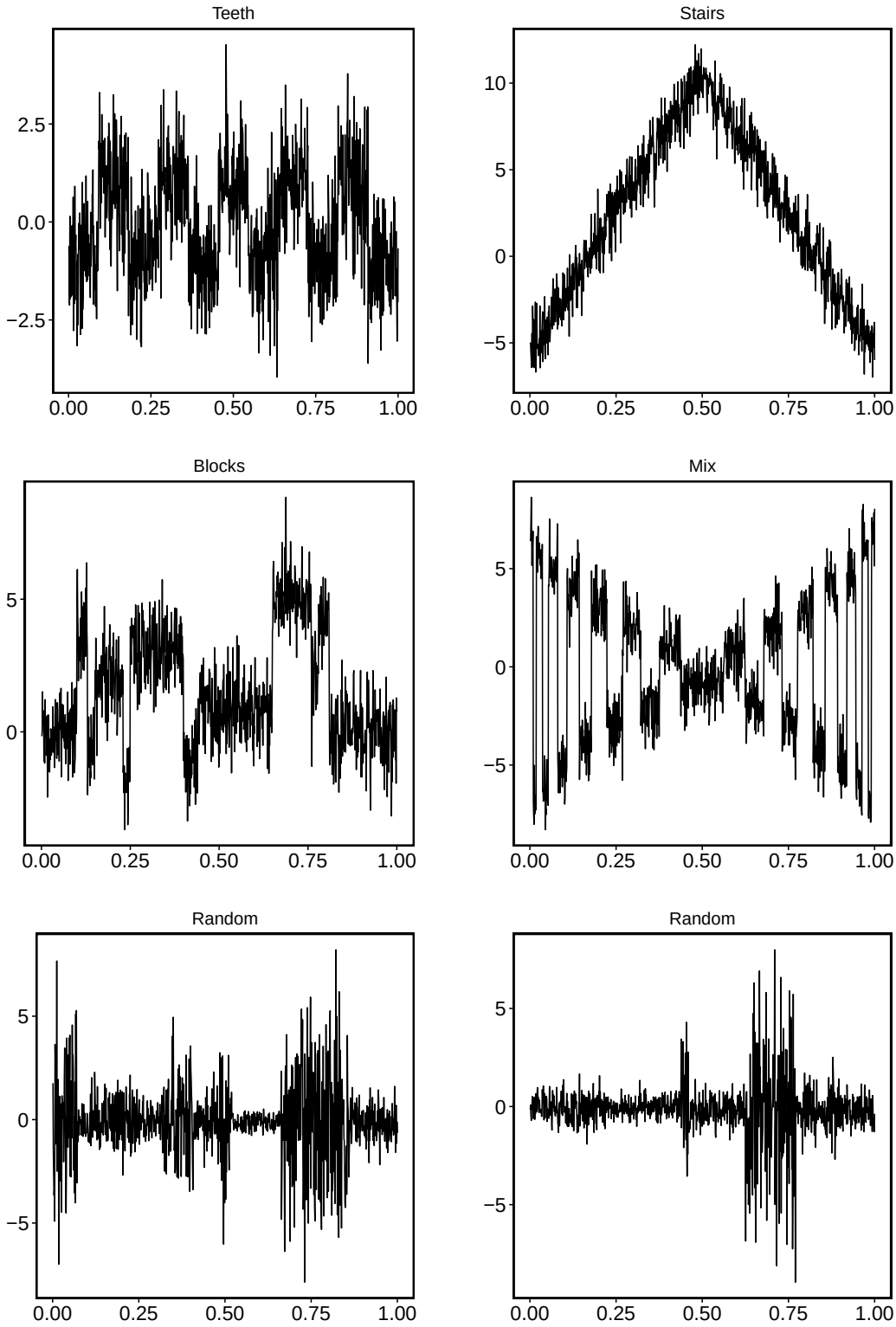


Figure 5.2: Example data-sets from the different test signals,

computational cost which is $\mathcal{O}(n)$ only when the number of changepoints is linear in the size of the data but in many situations the cost of PELT is $\mathcal{O}(n^2)$. We believe PELT will significantly benefit from parallelisation.

5.4.3 Detection Rate

We first of all want to look a the detection rate for different segment lengths, size of jumps and number of cores. For this initial analysis we will use the teeth and stairs signals because we can control the size of the jumps and the length of the segments. The two signals differ in the sense that the jumps in the teeth signal increase and decrease alternatively whereas in the jumps in the stairs signal all increase to a point and then decrease. Whilst the motivation for parallelisation comes from analysing large data, we study the statistical accuracy of the different methods on simulated data where n is relatively small. This is because it is easier to identify the characteristics of each methods, and what aspects of the data effect performance with these smaller data-sets.

We simulate 100 replications of the test signals with length 1000 and distances between the changes, s_i, 5, 10, 15, 20, 50 and 100. We also look at varying sizes of changes $\Delta\mu = 2, 3, 5, 10$ and use an even number of cores from 2 to 16. Finally we add some Gaussian noise to the signals, $N(0, 1)$.

Results

Figures 5.3 and 5.4 show the true and false positive detection rates for both the teeth and stairs signals using both SM1 and SM2. The results in this plot are for $s_i = 5, 20, 100$, further results for $s_i = 10, 15, 50$ can be found in Appendix E.

When the segment lengths are large SM2 outperforms SM1. This is also true for when the size of the jumps are large. However when the segment lengths and the jumps are small SM1 outperforms SM2 but it is clear that the performance decays as the number of cores increases. Due to the nature of the data, SM2 works better in

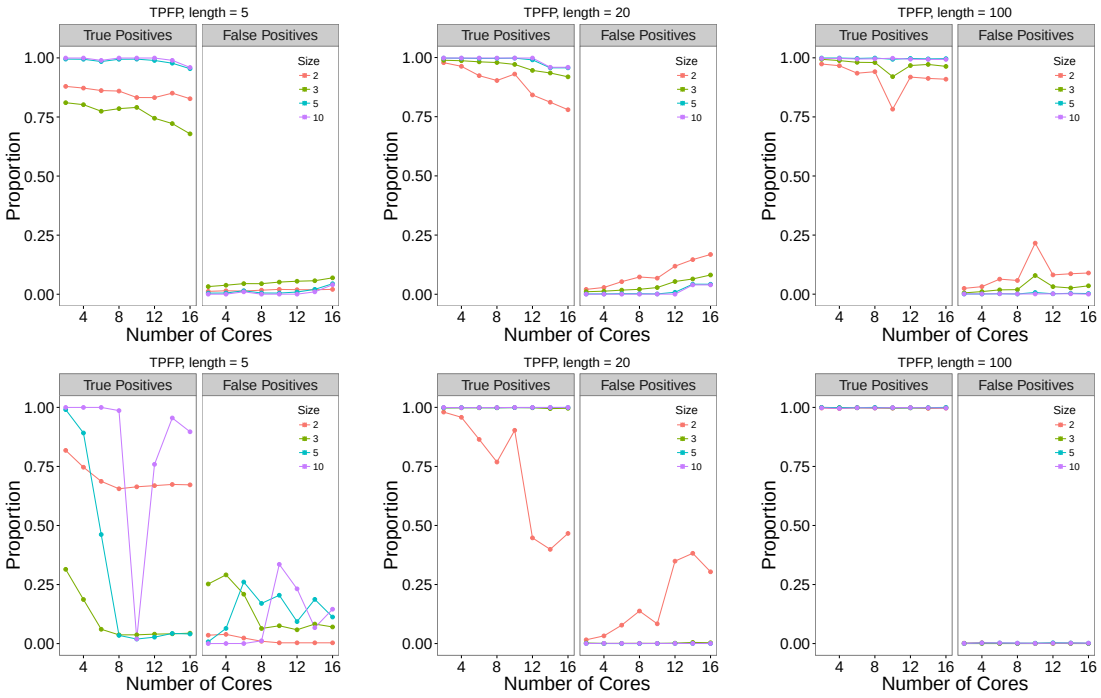


Figure 5.3: The TFDR for SM1 (top) and SM2 (bottom) on the teeth signal over a different number of cores compared to detecting the changes on one core. The plots left to right are different lengths of the segments: left - 5, centre - 20 and right - 100. The different coloured lines on each plot are different jump sizes.

the stairs signal than in the teeth signal when the segment lengths are small. This is due to the way the data is split over the cores in SM2. If there are more cores than data-points in a segment then some features of the data will be missed. This is more of an issue in the teeth data-set, where the data jumps up and then down, as it will end up missing one of these jumps and thus it will falsely look like 2 points on a core have come from the same segment whereas in reality there is a different segment between them.

In SM1 there is a spike at 100. This is a quirk in the method that is a result when the data is split on a true changepoint. Thus it falsely appears that this method is performing better than it actually is since the true changepoint is part of the set searched over in step 2 but this might only be because it was where the data was split. When $s_i = 100$ this spike looks like the method is not performing well when the mean is small but actually the method of detecting the changes over 1 core is sometimes

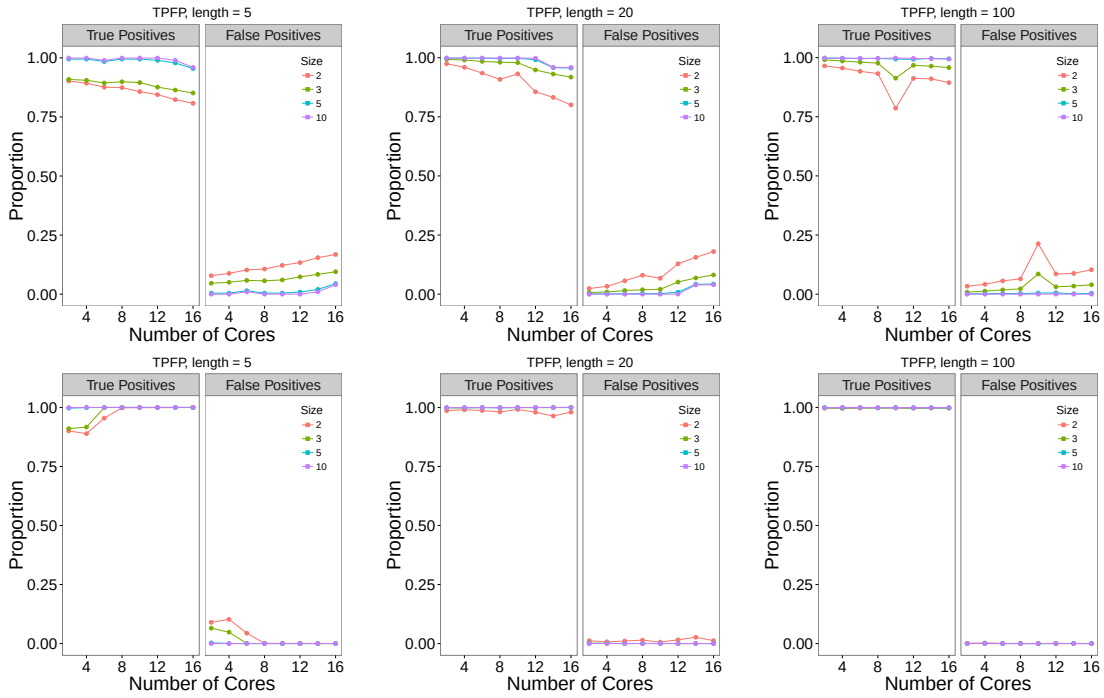


Figure 5.4: The TFDR for SM1 (top) and SM2 (bottom) on the stairs signal over a different number of cores compared to detecting the changes on one core. The plots left to right are different lengths of the segments: left - 5, centre - 20 and right - 100. The different coloured lines on each plot are different jump sizes.

detecting the changes 1 or 2 time-points from the truth and thus SM1 is artificially performing better.

Another quirk in the methods can be seen in the stairs signal with $s_i = 5$. The performance of SM2 appears to improve as the number of cores increases. This is because the jumps are very small and on a small number of cores the method misses some of the jumps. Whereas when there are more cores the size of the jump appears to be larger on each core so more true changepoints are detected.

Due to the nature of the equidistant segment lengths in the teeth and stairs signals we see some quirks in the data where the data is split over cores on a change. Figure 5.5 shows the performance of these methods on the blocks and mix signals where we have made the segment jumps to be of equal size. Here we can clearly see SM2 performs better than SM1 especially when the size of the jumps is large. The reason it does not appear to perform as well in the mix signal with small jumps than it does

in the blocks data is because the mix data signal has some smaller segments than the blocks signal.

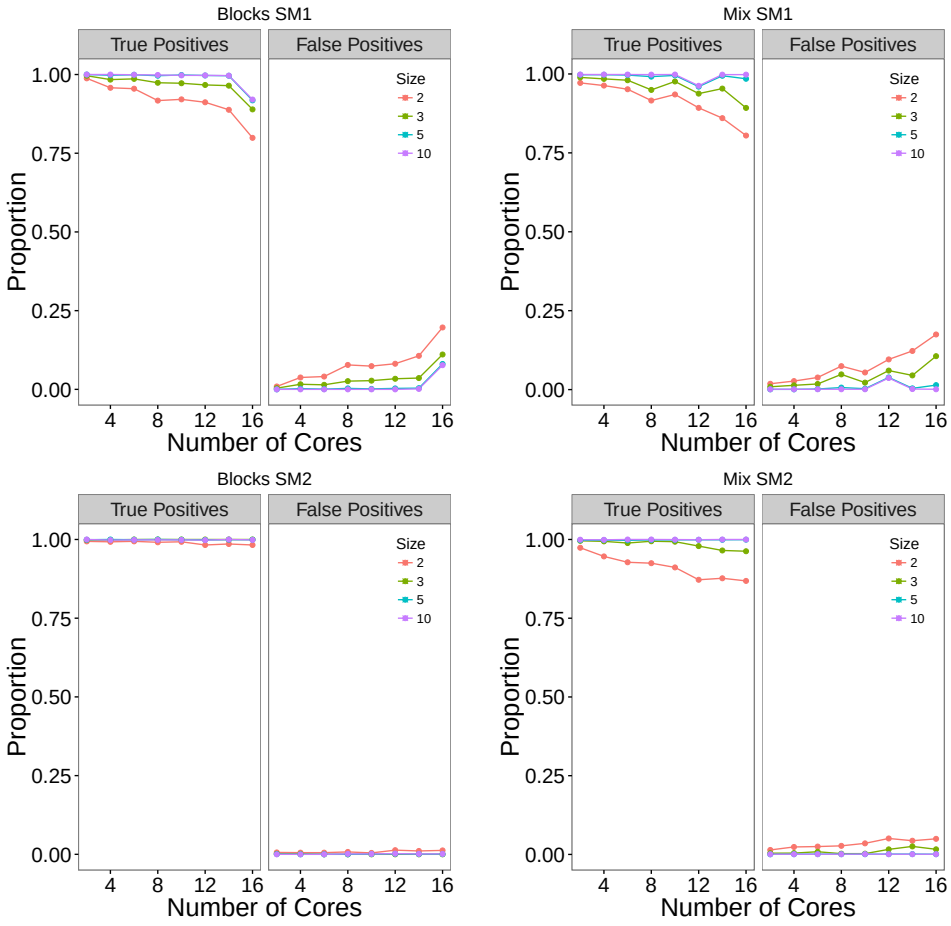


Figure 5.5: The TFDR for SM1 (top) and SM2 (bottom) for the blocks (left) and mix (right) signals with the same size jumps. The different coloured lines on each plot are different jump sizes.

Boundaries

In Section 5.3.4 we discussed the problem of missing changepoints when we split the data in SM1 especially if the change occurs near the boundaries. There are a couple of ways in which we can address this problem:

Fixed number of points The most obvious way to do this would be to choose a fixed number of points before and after the boundary however the number which this should be is not obvious in itself.

Use the detected changepoints before and after the boundary Another option would be to use the points in between the last detected changepoint in the segment before and the first detected changepoint in the segment after.

To investigate the boundary choice we use the above data with segment length 20 since the exact changepoint detection methods on one core perform consistently in this case for all sizes of jump but the proportion of true detected changes decreases at a different rate for each of the means as the number of cores increases. We look at boundaries with 0, 2, 5, 10, 15, 20, 25 and 30 points either side.

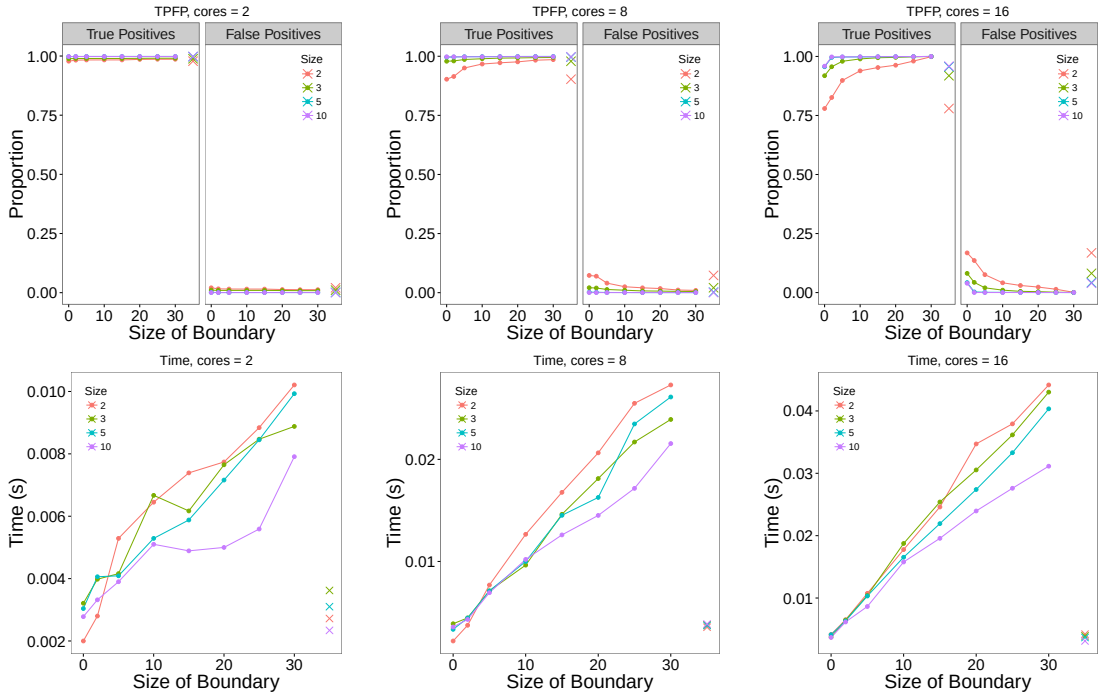


Figure 5.6: The results of using SM1 on the teeth signal with different number of points around the boundary. Top - true and false discovery rates and bottom - the computational time for step 2 of SM1. The crosses are the results from using an adaptive boundary.

Results

Figures 5.6 and 5.7 show the results for SM1 with different boundary lengths. Here we show the results when using 2, 8 and 16 cores. We tested this on number of cores from 2 to 16 and found similar results using the other numbers not shown

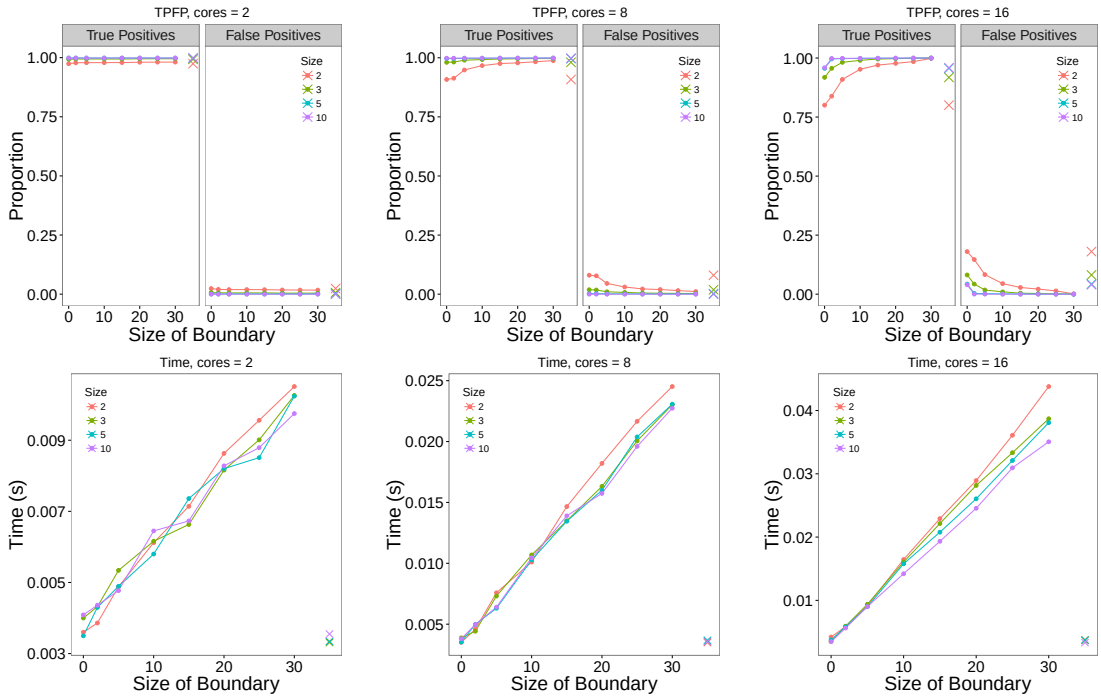


Figure 5.7: The results of using SM1 on the stairs signal with different number of points around the boundary. Top - true and false discovery rates and bottom - the computational time for step 2 of SM1. The crosses are the results from using an adaptive boundary.

here. The top plots show the TDR and FDR and we can see that as the boundary length increases the method does perform more accurately. This obviously comes at an increased computational cost. We can see from the bottom plots that the time for step 2 increases linearly with the more points we add around the boundary. The random fluctuations in the plot are just down to random fluctuations in the time using the computer cluster. Using the adaptive boundary lengths is computationally more efficient but it lacks the accuracy of using a fixed boundary length. When the data was split near but not on a changepoint then a false changepoint is sometimes detected 1 or 2 time-points away from the true change. Thus the true changepoint was then only 1 or 2 time-points away from the edge of the adaptive boundary and not detected in step 2. In this scenario the adaptive boundary performs poorly.

5.4.4 Speed

Up until now we have focussed on the performance of the methods in terms of the accuracy of the changepoints detected. We now turn our attention to the computational costs. We investigate the speed of the methods for increasing data lengths using a different number of cores. Additionally we investigate the times when the number of changepoints stays fixed as the data length increases and when the number of changepoints also increases.

Fixed number of changepoints

We first of all explore the times using the blocks and mix signals since the changes always appear at the same proportion of the data. That is, as the data length increases the number of time-points between the changes also increases. We simulate 100 replications of the test signals with length, $n = (1000, 5000, 10000, 15000, 20000, 50000, 75000, 250000)$ and we add some Gaussian noise, $N(0, 1)$ to the signals. We are interested in the time taken to run SM1 and SM2 over a different number of cores. For SM1 we use a boundary length of 20. The computational times are shown in Figure 5.8 and 5.9. We have plotted the smaller data lengths on the left and larger data lengths on the right on different scales. For both of the signals we can see the speed improvements of using multiple cores when the data length is large however using the maximum available is not always better. This is what we expect though from Amdahl's law (Amdahl, 1967). In all of the simulations the methods performed almost perfectly with TDR = 1, this was only slightly less when the number of cores was large.

Increasing number of changepoints

We now explore the times using the random signal. For this signal we specify the number of changepoints to be $n/1000$ and thus the number of changes increases with increasing data length. Again we explore the times of SM1 with boundary length 20

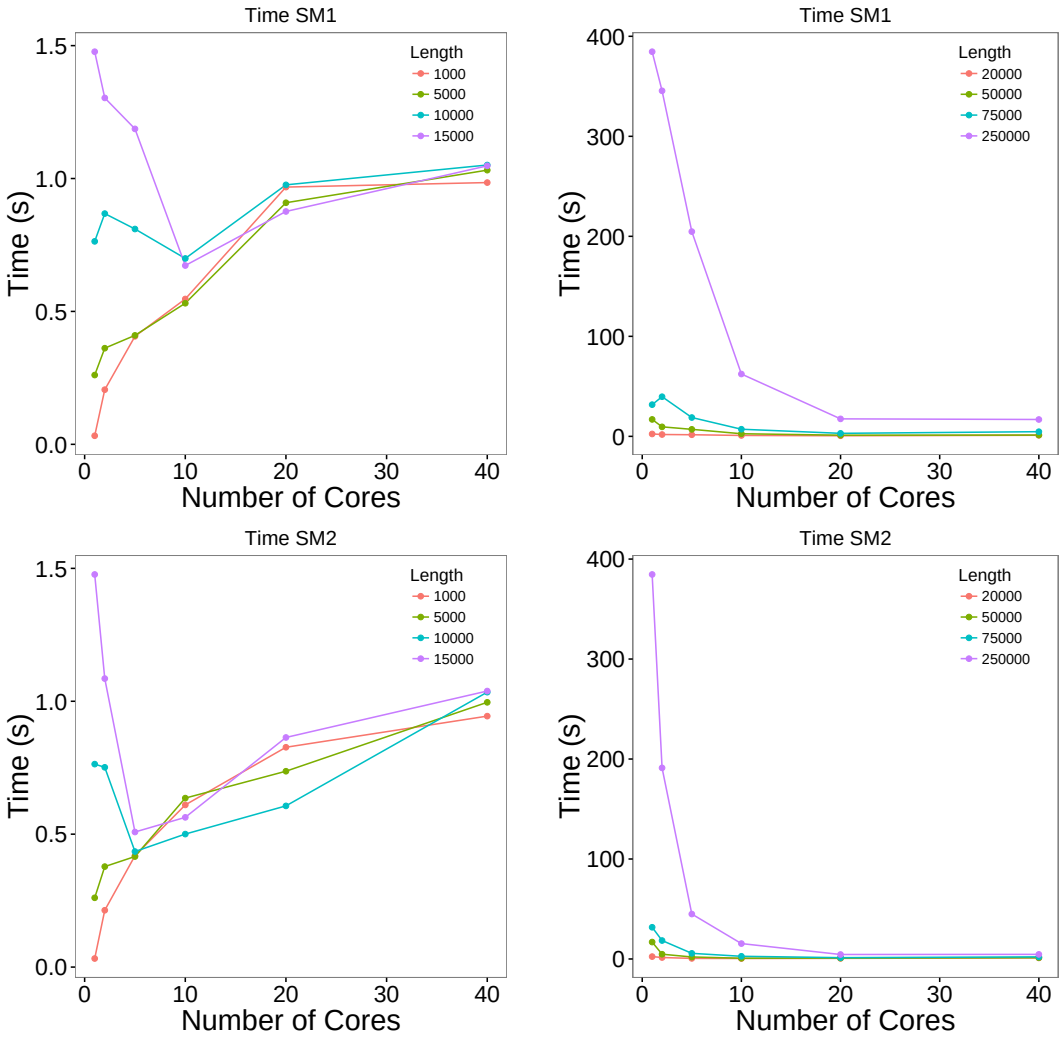


Figure 5.8: The CPU time for SM1 and SM2 on the blocks signal of different lengths, on the left are small data lengths and on the right are larger data lengths

and SM2. The results can be seen in Figure 5.10.

As in the simulations with a fixed number of changepoints, we see speed improvements using multiple cores in this scenario, especially when we have large data-sets. Using parallelisation is not as advantageous when the number of changepoints is linear since PELT performs more efficiently and has been shown to have a computational cost of $\mathcal{O}(n)$ (Killick et al., 2012). In this scenario the proportion of true changepoints detected using SM1 decreases when the number of cores increases for all data lengths (except $n = 1000$), this is due to the randomness of the data and the method failing to detect some of the smaller segments. Data length $n = 1000$ appears to perform

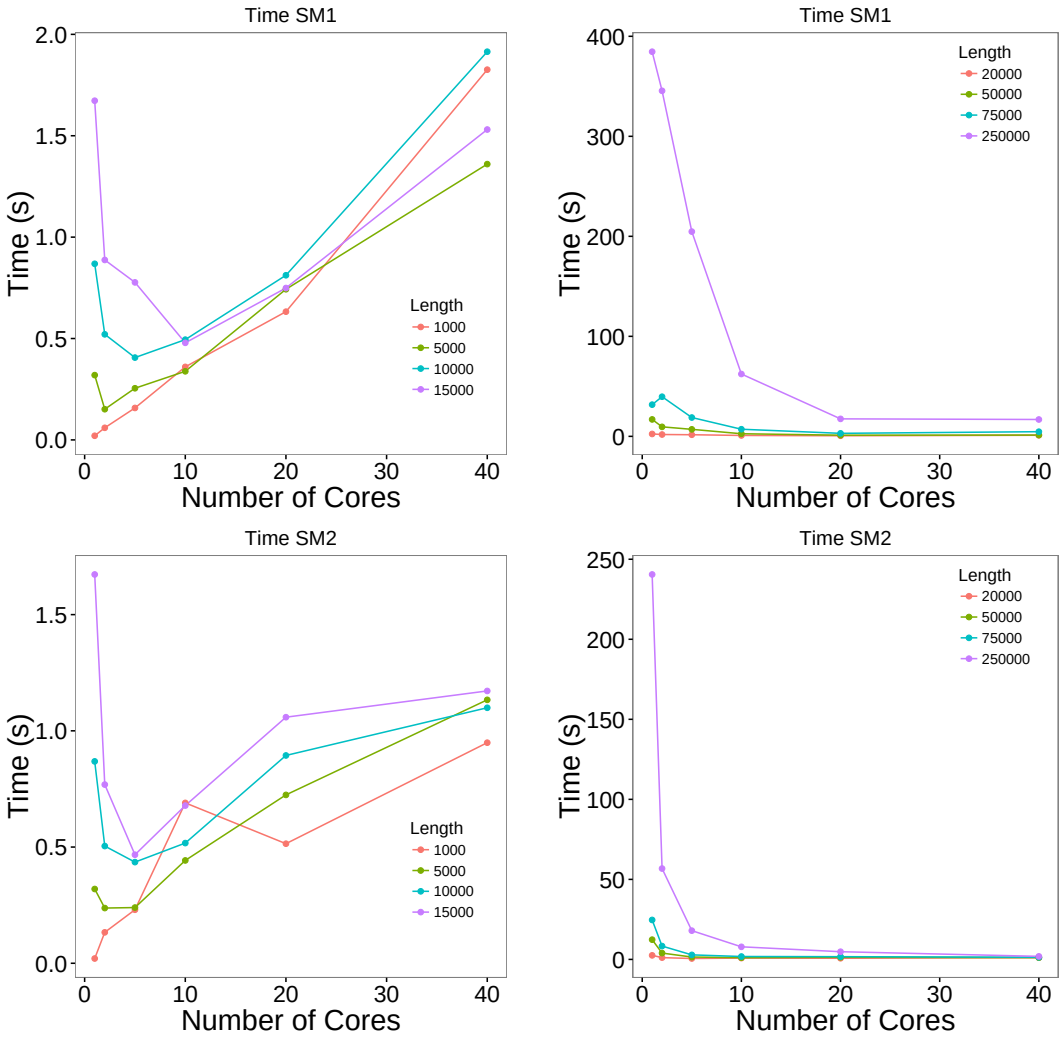


Figure 5.9: Computational time for SM1 and SM2 on the mix signal of different lengths, on the left are small data lengths and on the right are larger data lengths

better with increasing number of cores, however this is due to a large proportion of the data being considered in step 2 of the algorithm, since the data length is small so the data between the boundaries are small, thus SM1 performs more similarly to PELT. In comparison SM2 performs well across all number of cores and data lengths and in all cases is more accurate than SM1.

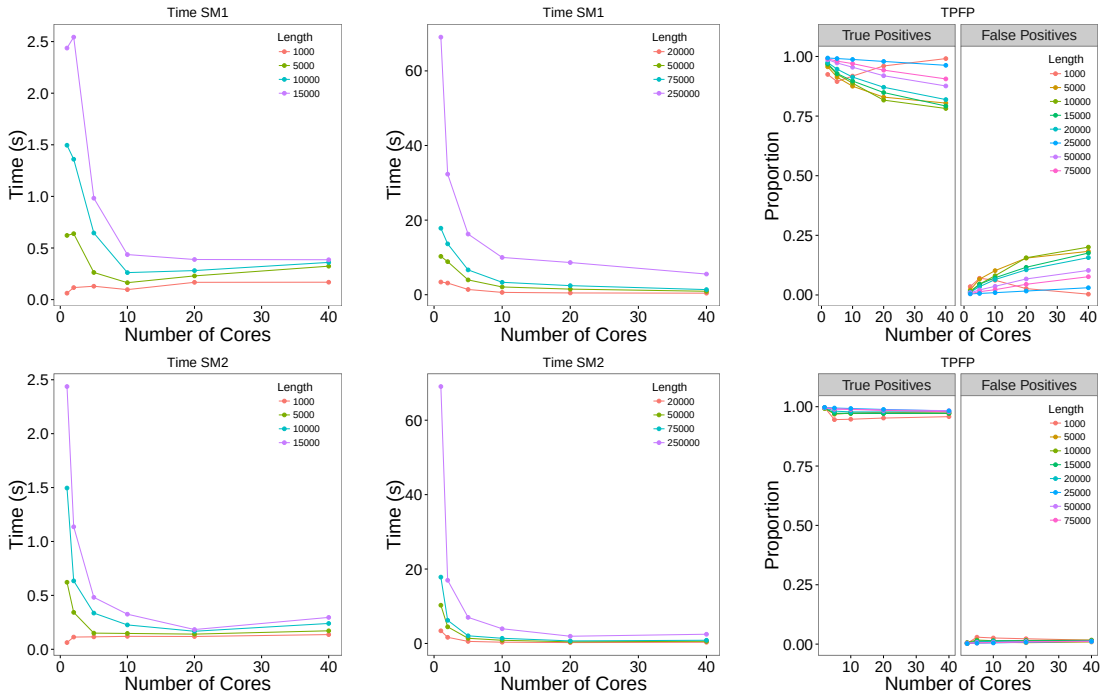


Figure 5.10: Computational time and TPFP for SM1 (top) and SM2 (bottom) on the random signals of different lengths with increasing number of changepoints. On the left are the times for small data lengths and in the middle are the times for the larger data lengths

5.5 Conclusion

In this chapter we have explored parallelisation for changepoint detection. We initially looked at Binary Segmentation type methods which are embarrassingly parallel. There was only a marginal improvement in the speed of Binary Segmentation when the data was large with lots of changes. Due to how quick the Binary Segmentation type algorithms are we found that the additional cost of communicating between cores did not make parallelisation feasible.

We then looked at changepoint approaches which use dynamic programming. Since each iteration of the algorithm requires calculations from the previous iterations, these methods cannot be easily parallelised. We proposed two split and merge approaches, similar to the method proposed by Song and Liang (2015) for Bayesian variable selection, to parallelise these methods. Both methods require the initial calculation of some summaries over the entire data-set. The first approach, SM1, splits the data

into equal size chunks in order and changepoints are detected on each subset of data simultaneously over multiple cores. The resulting changepoints are then collected together and a changepoint detection method is then run again but only allowing changes at the location of the changepoints detected in the first step. This approach will struggle to detect changepoints that occur near the boundary of one of the subsets of data. To overcome this issue we looked at various choices of boundary including a fixed number of points and an adaptive boundary length which takes the points from the last changepoint in the segment before the boundary to the first changepoint in the segment after. In the simulations we found that although the adaptive boundary length was more computationally efficient it did not perform as well as using a fixed boundary length with 20 or more points around the boundary.

Alternatively, we proposed another method which avoided the boundary issue of SM1. This new method, SM2, splits the data such that the first point goes to the first core, the second to the second core, and so on until all cores have one data-point and then the next data-point in the sequence goes to the first core. This is repeated until all of the data-points and shared over the cores.

We performed thorough analysis for many different data signals using SM1 and SM2. In the simulations we saw that SM1 performs better than SM2 when there are small segments. Due to the nature of how the data is split in SM2 it works best when there are a large number of points between changes. The benefit of SM2 is that it does not have the boundary issue that we need to deal with in SM1. Both methods provide a similar speed improvements with SM2 being slightly faster than SM1.

Chapter 6

Conclusions and Future Directions

In this thesis we have presented three different developments to current changepoint detection algorithms with the aim to extend their usage to a larger volume and variety of data-sets. The main framework used throughout this thesis was offline, univariate, multiple changepoint detection. In particular, we mainly focussed on methods which solved a penalised optimisation problem to find the optimal number and location of the changepoints simultaneously, using dynamic programming.

In the penalised optimisation approach the choice of penalty affects the performance of the changepoint detection method. If the penalty value is too large then we may miss some true changepoints, however if this value is too small we may end up falsely detecting changes. In Chapter 3 we developed a new algorithm, CROPS (Changepoints over a Range of PenaltieS), which finds the optimal solution over a range of penalties in a continuous range. We applied CROPS to detect genomic regions that interact through the folding and 3-D structure of a chromosome. We then used an "elbow" approach suggested by Lavielle (2005) to find the best segmentation. The CROPS algorithm is an alternative to the Segment Neighbourhood Search (SN, Auger and Lawrence, 1989) which outputs the optimal segmentations for a range of number of changepoints. In simulations we showed that CROPS is computationally faster than SN. CROPS does not output all of the optimal segmentations that are

found using SN. The segmentations which CROPS does not recover corresponds to ones where adding an extra changepoint leads to a larger change in cost than removing a changepoint.

In Chapter 4 we turned our attention to the cost function used in the optimisation problem. Traditionally the cost functions used, such as the log-likelihood, require knowledge of the distribution of the data and the type of change. We proposed a new algorithm ED-PELT (PELT with an Empirical Distribution cost function) which uses a cost function based on the empirical distribution of the data. This new cost function was suggested by Zou et al. (2014) but to speed up the calculation of the segment costs we use an approximation of the integral in Zou et al. (2014). This approach is nonparametric and thus can be applied to a large variety of data-sets since we do not need to know what the distribution of the data is or what the type of changes present are. We applied ED-PELT to heart-rate data recorded during a period of physical activity and show that ED-PELT gave a better segmentation than using a cost function which assumed the data was normally distributed.

In Chapter 5 we looked at using high-performance computers and parallel algorithms to speed up the computation of using dynamic programming methods. Popular dynamic programming methods such as Optimal Partitioning and Segment Neighbourhood Search have a computational cost that is quadratic in the length of the data. There are pruning methods which reduce this computational cost however these costs are still large for huge data-sets. In Chapter 5 we looked at ways to split the data over multiple cores and then combine the results whilst still conserving most of the accuracy that we had when we only used one core. Our split and merge approaches are similar to the approach in Song and Liang (2015) for Bayesian variable selection in the sense that we split the data over multiple cores and detected the changes on each core, we then ran changepoint detection using these changes as the candidate changepoint locations to get an overall solution. In this Chapter we comprehensively looked a the performance of our split and merge approaches across many different

scenarios.

The CROPS algorithm proposed in Chapter 3 has been implemented in the `changepoint` R package (Killick et al., 2014). Similarly the ED-PELT algorithm has been implements in the `changepoint.np` algorithm (Haynes, 2016). Code to run the parallel methods in Chapter 5 is available at https://github.com/KayleaHaynes/changepoint.parallel. Details on all of these are provided in the Appendices.

6.1 Future Directions

Below are some suggestions on how the work presented in this thesis can be extended in the future. These are:

1. Apply a probabilistic pruning approach to the nonparametric cost function in Chapter 4.

2. Explore alternative nonparametric cost functions to use with PELT.

3. Expand the parallelisation methods in Chapter 5 to parallelise multivariate changepoint detection methods.

6.1.1 Probabilistic Pruning

In Chapter 4 we used the inequality based pruning method, PELT, with a nonparametric cost function based on the empirical cumulative distribution of the data. It was suggested by one of the reviewers when submitting this paper to Statistics and Computing that an alternative method would be to use the probabilistic pruning approach, cp3o (Changepoints via Probabilistic Pruned Objects; James and Matteson, 2015). Probabilistic pruning can be applied to a large number of goodness of fit methods and can generate all of the optimal segmentations with differing number of

changepoints. The pruning procedure is similar in the set up to the pruning in PELT. Using the notation from Chapter 4, if we define

$$Q_{\text{cp3o}} = \min_{m,\tau_{1:m}} \left\{ \sum_{i=1}^{m+1} \mathcal{C}_K(x_{(\tau_{i-1}+1):\tau_i}) + \xi_n \right\}, \tag{6.1}$$

then with probability of at least $1-\epsilon$, u can never be the most recent changepoint prior to T where $u < v < T$. That is

$$\mathrm{P}(Q_{\text{cp3o}}(x_{1:u}, \xi_n) + \mathcal{C}_K(x_{(u+1):v}) > Q_{\text{cp3o}}(x_{1:v}, \xi_n)) \geq 1 - \epsilon. \tag{6.2}$$

For the consistency of cp3o to hold using the cost function suggested in (4.7) some assumptions about the goodness of fit test are required. Firstly, assumption 5 in James and Matteson (2015) requires that under the single changepoint detection scenario the optimal value of the goodness of fit is attained when the estimated changepoint $\hat{\tau}$ coincides with the true changepoint τ. Given a suitably chosen K in our cost function this should hold.

Secondly, assumption 7 in James and Matteson (2015) requires for all $u < v < T$

$$\mathcal{C}_K(x_{u+1:T}) \geq \mathcal{C}_K(x_{u+1:v}) + \mathcal{C}_K(x_{v+1:T}). \tag{6.3}$$

This assumption is shown to hold in the proof of Theorem 4.3.2 in Chapter 4. In terms of performance James and Matteson (2015) compare cp3o with an Energy statistic to PELT with a change in mean for a couple of different scenarios. PELT performed worse in terms of accuracy due to the misspecification of the model but it was the faster algorithm. Given the same cost function it would be interesting to compare the two methods in terms of accuracy and to see if PELT is still computationally quicker as this may have been down to the calculation of the cost functions.

6.1.2 Nonparametric Cost Functions

The cp3o method discussed above was originally used with a divergence measure based on Euclidean distances (Matteson and James, 2014). Matteson and James (2014) used a hierarchical clustering method to detect the changes and also showed how to approximate the speed up of the calculations. The cost function used is

$$\begin{aligned}\mathcal{C}(Y_n, Z_m; \alpha) &= \frac{2}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}|Y_i - Z_j|^\alpha \\ &- \binom{n}{2}^{-1}\sum_{1\le i<j\le n}|Y_i - Y_j|^\alpha - \binom{m}{2}^{-1}\sum_{1\le i<j\le m}|Z_i - Z_j|^\alpha, \quad (6.4)\end{aligned}$$

where $Y_n = \{x_a, x_{a+1}, ..., x_{a+n-1}\}$ and $Z_m = \{x_a + 1, x_{a+n+1}, ..., x_{a+n+m-1}\}$. In order to use PELT the following property must hold

$$\mathcal{C}(x_{u+1:T}) \geq \mathcal{C}(x_{u+1:v}) + \mathcal{C}(x_{v+1:T}) \quad (6.5)$$

for all $u < v < T$. Equation (6.5) has been shown to hold with the cost function in (6.4) (James and Matteson, 2015, proposition 7) and thus can be used in the PELT algorithm.

Zou et al. (2014) show that this test statistic performs better than their method NMCD for changes in the scale when errors have either a t-distribution or chi-squared. However this cost function can only be used when there are changes in the first two moments. Such as in Section 6.1.1 if would be interesting to compare PELT and cp3o using this cost function as well as comparing to PELT with the empirical distribution cost.

6.1.3 Parallelising Multivariate Methods

Parallelisation would lend itself nicely to improve the computational costs of multivariate changepoints detection, particularly when considering subset multivariate

changepoints; a changepoint occurs only in a subset of the variables. Pickering (2015) proposes a method, SMOP, which first considers costs for a multivariate time-series using the notion of *changepoint vectors*. If we let c_t^j be the location of the most recent changepoint in variable j up to and including time-point t then the changepoint vector at time t is the vector of all the most recent changepoints across all of the variables.

The multivariate cost can then be defined as follows. If we let $\mathcal{D}_j(\cdot)$ be a generic additive cost function, such as the negative log-likelihood, for each variable $j = 1, ..., p$, and let $q_{\tau_K} = \sum_{j=1}^{p} \mathbf{1}(\tau_K^j = \tau_K)$ denote the number of dimensions affected by a change, then the multivariate penalised cost is

$$\sum_{k=1}^{m+1} \left[\sum_{j=1}^{p} \left\{ \mathbf{1}(c_{\tau_k}^j = \tau_k) \mathcal{D}_j(y_{(c_{\tau_{k-1}}^j+1):c_{\tau_k}^j})^j + \alpha g(q_{\tau_k}) \right\} \right] + \beta f(m). \tag{6.6}$$

Here $\alpha g(q_{\tau_k})$ is a penalty to avoid over-fitting the number of variables affected by the kth changepoint, and $\beta f(m)$ is the penalty term to avoid over-fitting the number of changepoints.

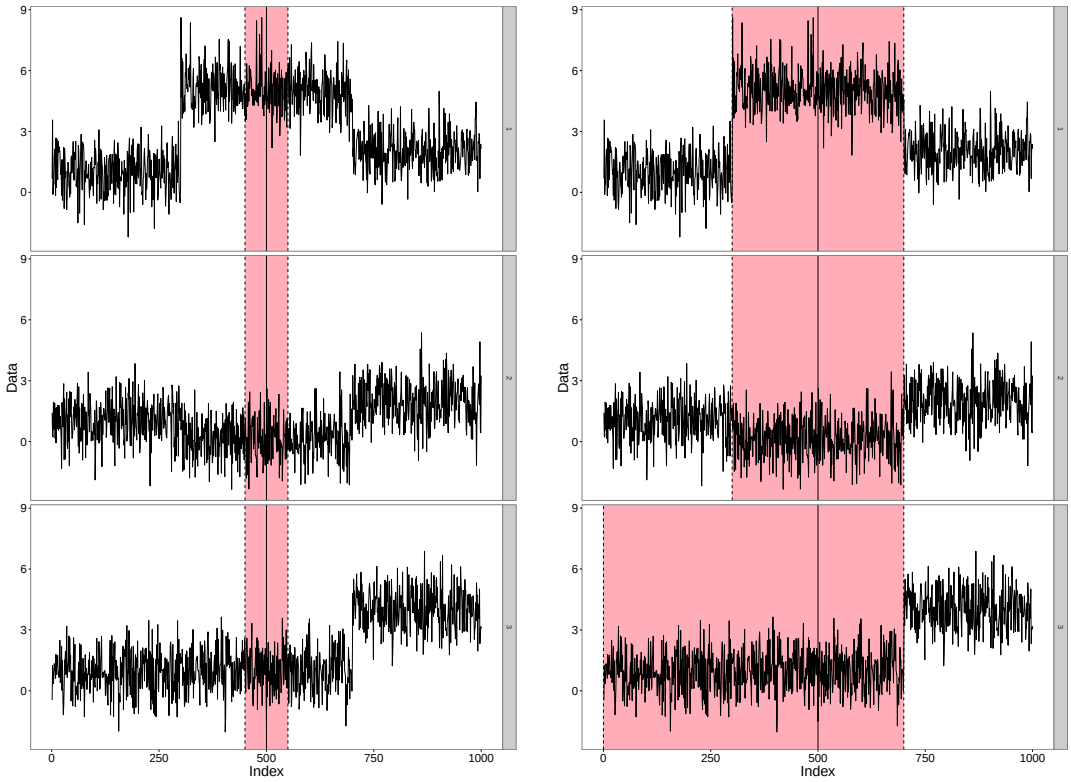
To find the optimal location of the changepoints, Pickering (2015) then used a dynamic programming algorithm similar to the univariate Optimal Partitioning method (Jackson et al., 2005). If p is the number of dimensions and n is the length of the data this has a computational cost $\mathcal{O}(pn^{2p})$. Unlike in the univariate setting, Pickering (2015) shows that pruning methods such as PELT (Killick et al., 2012) are not practically viable in the multivariate case due to the additional storage and calculations required.

SMOP is computationally expensive even for relatively small values of n. In order to improve the speed Pickering (2015) suggests a couple of approximations in their alternative algorithm, A-SMOP, to reduce the number of potential changepoint locations and variables affect which SMOP considers. The first approximation runs univariate PELT on each variable to get a candidate set of locations from each variable to use in SMOP. The second approximation uses two window arguments to reduce

the number of affected variables for each changepoint.

It should be possible to parallelise this changepoint detection method. Summaries of each variable can be calculated beforehand and then used to calculate the corresponding $\mathcal{D}(\cdot)$ costs similar to the univariate case as shown in Section 5.3.1. In Chapter 5 we proposed two split and merge approaches for parallelising changepoint algorithms. SM1 is where the data is split in chunks over the cores with the first l points going to the first core, the next l going to the second core etc. SM2 is there the data is split such that the first data-point goes to the first core, the second data-point goes to the second core etc. I believe it should be possible to use SM2 in this scenario without too much divergence from the univariate set-up. SM1 will be more difficult, but not impossible, to implement due to the boundary values. If we take 50 points either side of the boundary, Figure 6.1a shows the changepoint vectors that would be considered had we looked at a fully multivariate scenario in an example with 3 dimensions and 2 cores. This example is simple since we are allowing changepoints to occur in all variables. The subset multivariate scenario is much more challenging. It may be easier to use an adaptive boundary in this scenario. Figure 6.1b shows an example of the changepoint vectors we would consider using the changepoints detected in each variable before and after the boundary.

Figure 6.2 shows the challenges of using a fixed boundary. Since we are allowing changes to not occur simultaneously over all variables then it is not as easy as just looking at the points either side of the boundary in all variables. The pink shaded region shows the changepoints in one dimension that would be considered when we have a boundary length of 50 either side of the boundary. The grey shaded regions show the changepoint vectors we would have to consider in each case. Here we are assuming all changepoints in the other dimensions (grey region) can only appear before or at the same location of changepoints of the dimension we are interested in (pink region). The main challenge of this approach will be keeping track of all of the possible changepoint vectors at each time step.



(a) (b)

Figure 6.1: The shaded regions show the changepoint vectors that will be considered around the boundary. (a) Is the fully multivariate case and (b) is the subset multivariate case using an adaptive boundary.

It would be very interesting in the future to look into whether using multi-cores allows SMOP to become a viable method for subset multivariate method without the need to approximate it.

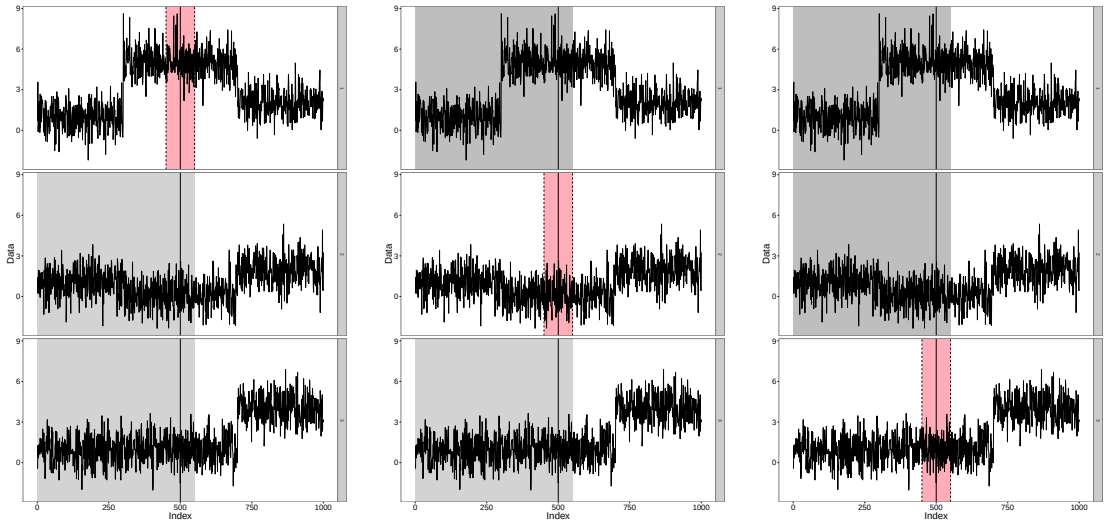


Figure 6.2: An illustration of the challenges of using a fixed boundary length.

Appendix A

Supplementary Material for Chapter 3

A.1 Pseudo-code for PELT

Algorithm 2: PELT

input : A data-set of the form $y_{1:n} = (y_1, y_2, ..., y_n)$;
A cost function $\mathcal{C}(\cdot)$ dependent on the data;
A penalty constant β, and a constant K that satisfies the condition for PELT for all $s < t < T$.

output: Details of the optimal segmentation of $y_{1:t}$ for $t = 1, \ldots, n$.

Let cp(0) = 0, rescp(0) = 0, $F(0) = 0$, $m(0) = 0$ *and* $R_1 = \{0\}$;

for $t \in \{1, ..., n\}$ **do**

1. Calculate $F(t) = \min_{s \in R_t}[F(s) + \mathcal{C}(y_{(s+1):t}) + \beta]$;
2. Let $cp(t) = \arg\min_{s \in R_t}\{[F(s) + \mathcal{C}(y_{(s+1):t}) + \beta]\}$;
3. Let $m(t) = m(cp(t)) + 1$;
4. Set $rescp(t) = [rescp(cp(t)), cp(t)]$.;
5. Set $R_{t+1} = \{s \in R_t : F(s) + \mathcal{C}(y_{(s+1):t}) < F(t)\}$.

end

return : $rescp(n)$: the changepoints in the optimal segmentation of $y_{1:n}$; and for $t = 1, \ldots, n$;
$cp(t)$: the most recent changepoint in the optimal segmentation of $y_{1:t}$;
$m(t)$: the number of changepoints in the optimal segmentation of $y_{1:t}$;
$F(t)$: the optimal cost value of the optimal segmentation of $y_{1:t}$.

A.2 Proof of Therem 3.1

Proof. To simplify notation, write $m_0 = m(\beta_0)$ and $m_1 = m(\beta_1)$. Part (1) follows immediately from the fact that $m(\beta)$ is a decreasing function.

For part (2), note that as $m(\beta)$ is decreasing, then $m(\beta)$ will be equal to either m_0 or m_1 for all $\beta \in [\beta_0, \beta_1]$. Using (3.1), to find the interval of values for which $m(\beta) = m_0$ we need to find the values of β for which $P_{m_0}(\beta) < P_{m_1}(\beta)$. The value β_{int} is just the solution to $P_{m_0}(\beta) = P_{m_1}(\beta)$. This gives the required result.

For part (3), first note that as $m(\beta)$ is decreasing, then as $m(\beta_{int}) = m_1$ we must have $m(\beta) = m_1$ for all $\beta \in [\beta_{int}, \beta_1]$. Thus we only need to show that for any m with $m_1 < m < m_0$ and for all $\beta \in [\beta_0, \beta_{int}]$,

$$Q_m(y_{1:n}) + m\beta \geq Q_{m_0}(y_{1:n}) + m_0\beta.$$

We show this by contradiction. Firstly assume there exists an m with $m_1 < m < m_0$ and a $\beta \in [\beta_0, \beta_{int}]$ such that

$$Q_m(y_{1:n}) + m\beta < Q_{m_1}(y_{1:n}) + m_0\beta.$$

As $m < m_0$ and $\beta \leq \beta_{int}$, this implies

$$Q_m(y_{1:n}) + m\beta_{int} < Q_{m_0}(y_{1:n}) + m_0\beta_{int},$$

and by definition of β_{int} we then have

$$Q_m(y_{1:n}) + m\beta_{int} < Q_{m_1}(y_{1:n}) + m_1\beta_{int}.$$

This then contradicts the condition of part (3) of the theorem, namely that a segmentation with m_1 changepoints is optimal for the penalty β_{int}. □

A.3 Proof of Theorem 3.2

Proof. The proof for part (1) is trivial since we need to run CPD twice, using both β_0 and β_1.

For the proof of part (2) define $N(m_0, m_1)$ as the maximum (over data-sets) of the number of further runs of CPD needed to find all the optimal segmentations in an interval of β, given we have run CPD at the lower and upper endpoints of the interval and these have produced segmentations with m_0 and m_1 changepoints respectively. As we have run CPD twice, to prove the theorem we need to show that

$$N(m_0, m_1) \leq m_0 - m_1 - 1. \tag{A.1}$$

Firstly, if $m_0 - m_1 = 1$ then $N(m_0, m_1) = 0$, which satisfies (A.1).

Now we proceed by induction. For an integer $l > 1$ assume that if $m_0 - m_1 < l$ then (A.1) holds. We need to show that this implies that (A.1) holds for $m_0 - m_1 = l$.

In this case our first step is to run PELT at the intersection, β_{int}. In the worst case scenario we find that $m(\beta_{int}) \neq m_1$ (and hence $m(\beta_{int}) \neq m_0$ as segmentations with m_0 and m_1 changepoints have the same penalised cost for penalty value β_{int}). We then need to consider the sub-intervals below and above β_{int} separately. Since $m(\beta)$ decreases as β increases $m_0 - m(\beta_{int}) < l$ and $m(\beta_{int}) - m_1 < l$. Therefore

$$\begin{aligned} N(m_0, m_1) &= 1 + N(m_0, m(\beta_{int})) + N(m(\beta_{int}), m_1) \\ &\leq 1 + [m_0 - m(\beta_{int}) - 1] + [m(\beta_{int}) - m_1 - 1] \\ &= 1 + m_0 - m_1 - 2 \\ &= m_0 - m_1 - 1. \end{aligned}$$

which satisfies (A.1) as required. □

A.4 Further Simulations: Change in Mean

In the main manuscript we looked at a change in mean model with a fixed number of changepoints and compared pDPA to CROPS with FPOP. Here we look at the results when the number of changepoints increases sublinearly and linearly with the number of data-points.

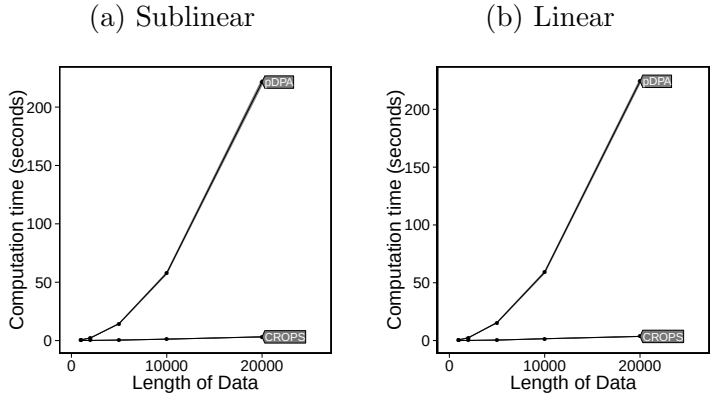


Figure A.1: Changes in mean CPU cost using pDPA and CROPS with FPOP. (a) Sublinear changepoints ($m = \sqrt{n}/4$) and (b) linear changepoints ($m = n/100$).

A.5 Further Simulations: Change in mean and variance for normal model

Similarly we revisit the model with a change in mean and variance to compare Segment Neighbourhood with CROPS with PELT and PELT with the recycled calculations for data-sets with a sublinear and linear number of changepoints in respect to data length. Figure A.2 shows the CPU cost for the 3 methods. We can then use this data and CROPS with PELT to explore penalty choice when we have a different number of changes. These results can be seen in Figure A.3.

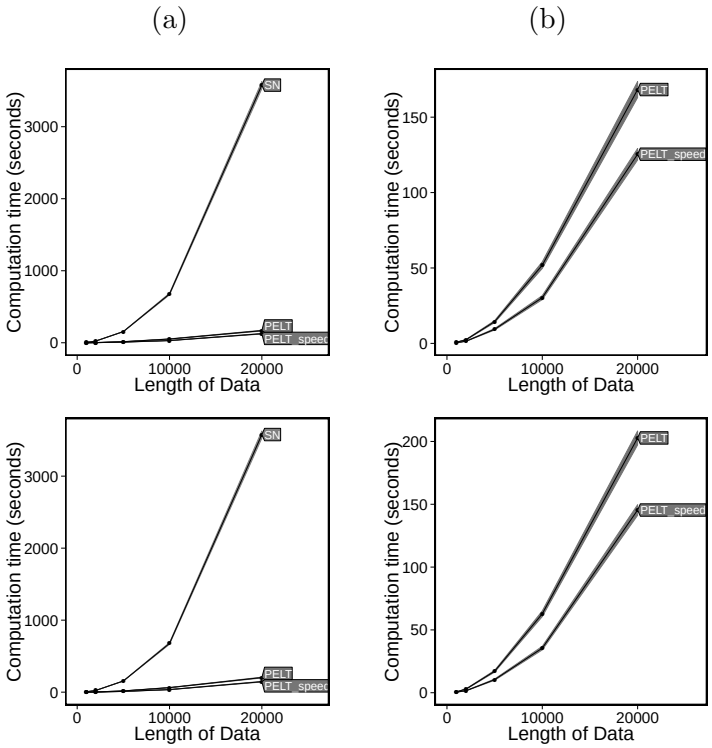


Figure A.2: (a) CPU cost using SN, CROPS with PELT and CROPS with PELT with the speed improvements. (b) A close up of PELT and PELT with the speed improvements. The top row is sublinear changepoints ($m = \sqrt{n}/4$) and the bottom row is linear changepoints ($m = n/100$).

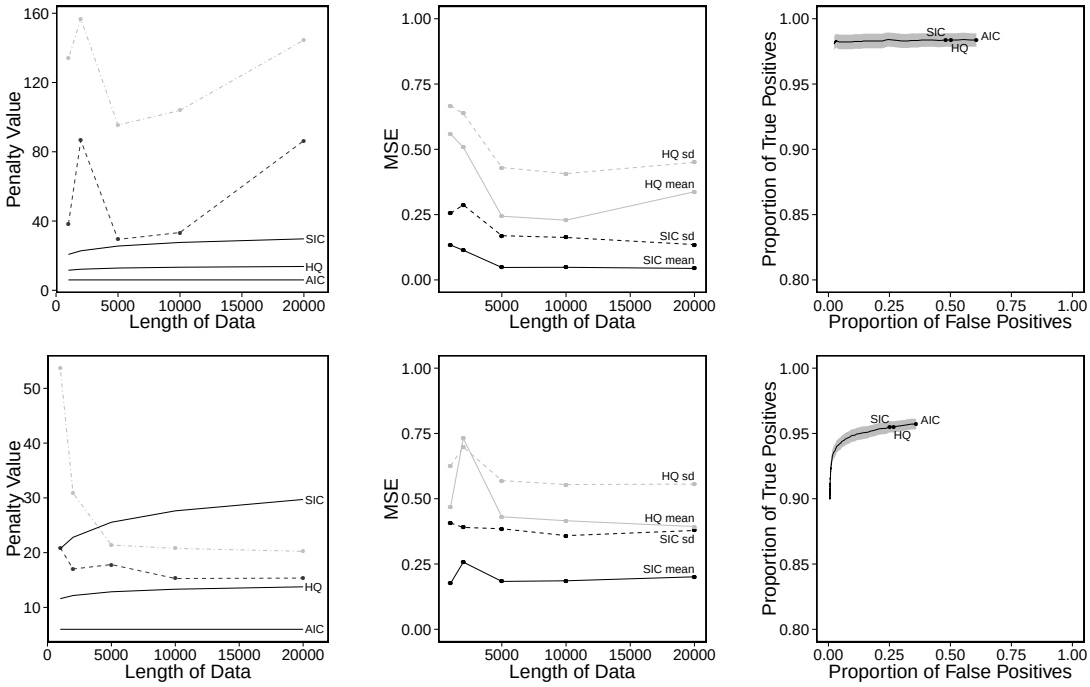


Figure A.3: Results for the true model. Left: Average minimum (black, dashed) and maximum (grey, dot-dashed) optimal penalty values in comparison to popular penalty terms in the literature. Solid lines from top to bottom are the SIC, Hannan-Quinn and AIC penalty values. Middle: MSE for the mean (solid) and the standard deviation (dashed) when different penalty terms are used. Right: Proportion of true positives against the proportion of false positives for $n = 10,000$. The top row is sublinear changepoints ($m = \sqrt{n}/4$) and the bottom row is linear changepoints ($m = n/100$).

A.6 Further Simulations: Change in Mean and Variance for the Mis-specified Model

We can also look at the situation where we have a mis-specified model with a sublinear and linear number of changes. The range of optimal penalty values and true and false positives found using different common penalty terms are shown in Figure A.4

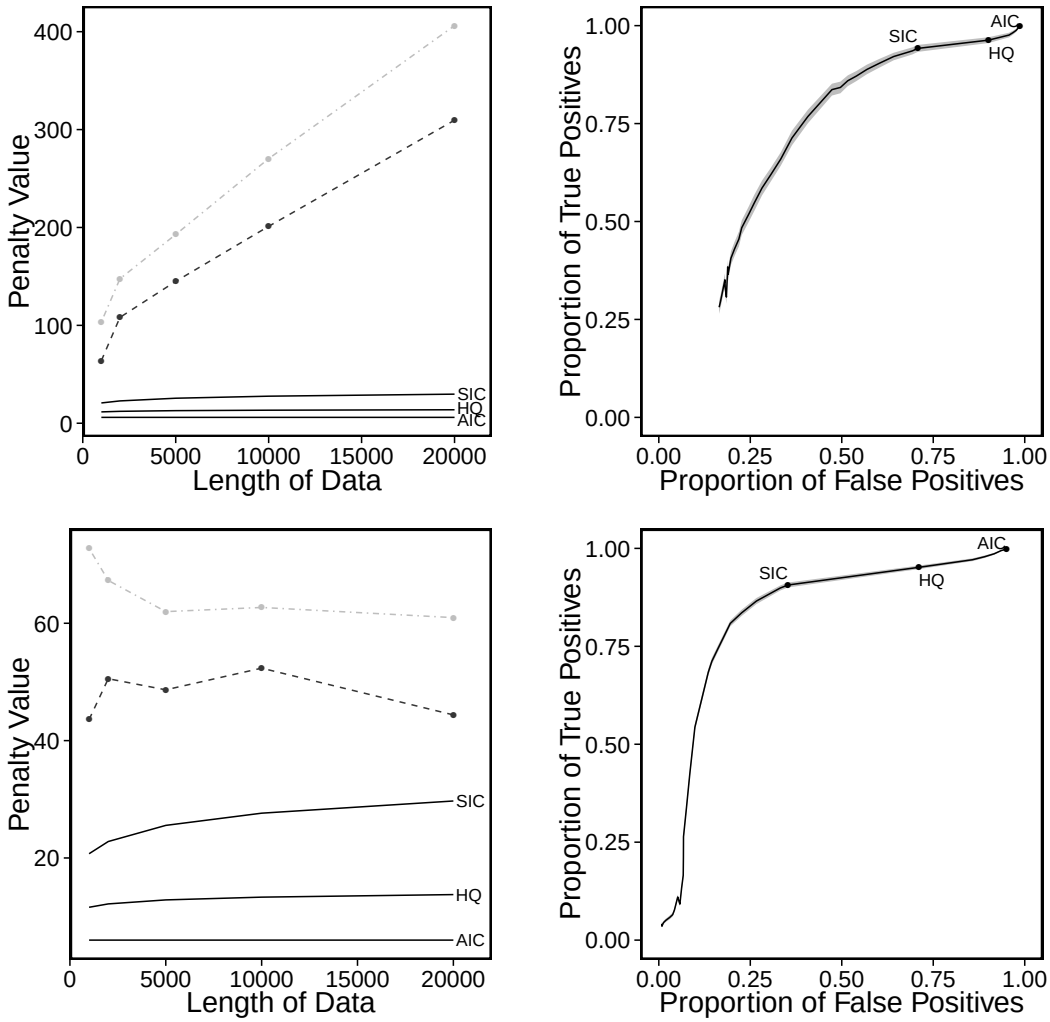


Figure A.4: Results for the mis-specified model scenario. Left: Average minimum (black, dashed) and maximum (grey, dot-dashed) optimal penalty values in comparison to popular penalty terms in the literature. Right: Proportion of true positives against the proportion of false positives for $n = 10,000$. The top row is sublinear changepoints ($m = \sqrt{n}/4$) and the bottom row is linear changepoints ($m = n/100$).

A.7 Further regions where we have discrepancies in the Hi-C example

In the real data section of the main manuscript we showed regions of the chromosome where was have detected different segmentations with the different penalty terms. Below are a further 3 regions.

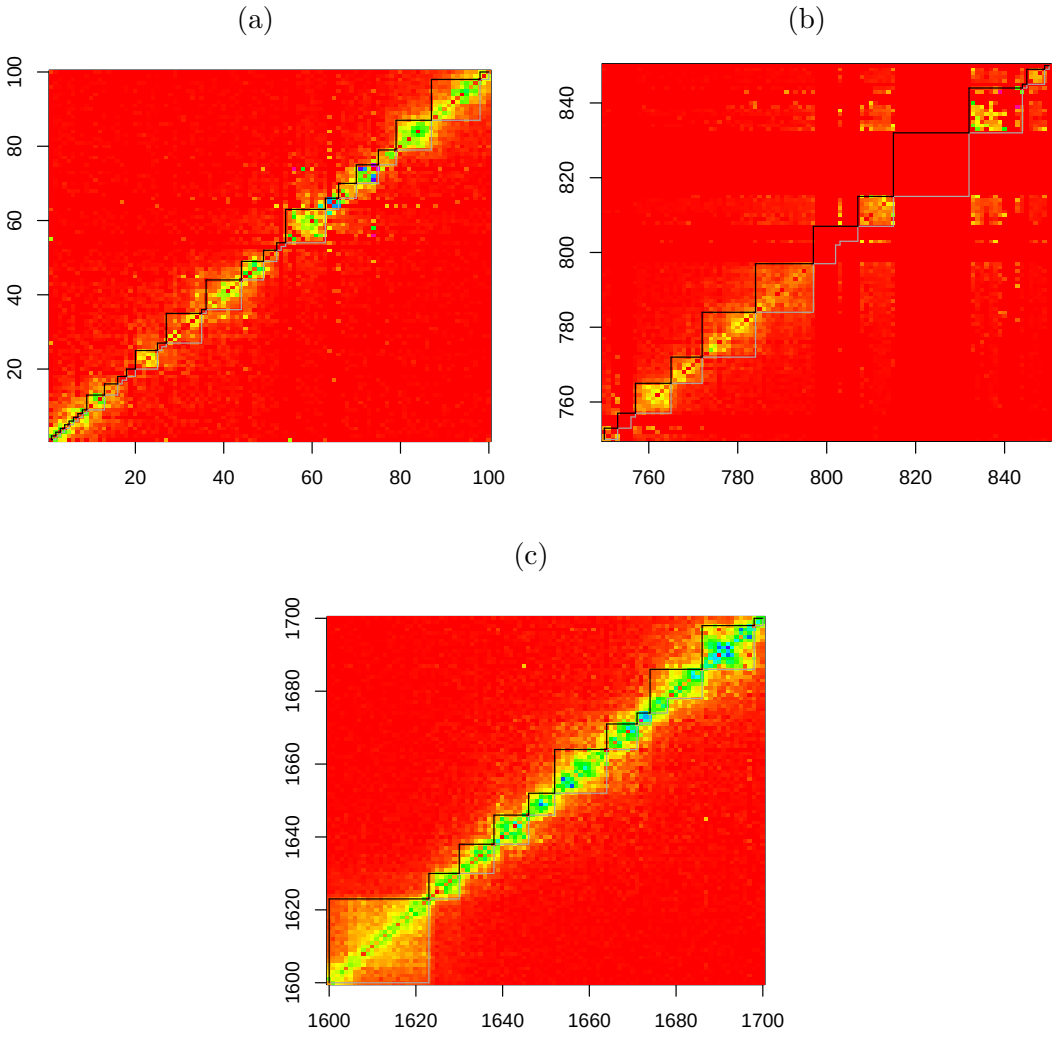


Figure A.5: Close up different regions, the black line is the segmentation using our optimal β and the grey line is the segmentation using $\beta = 0$ and $\beta = \text{SIC}$.

Appendix B

The CROPS Algorithm in the `changepoint` R Package

The CROPS algorithm proposed in Chapter 3 has been implemented within the `changepoint` R package (Killick et al., 2014). The usage is similar for all of the functions within this package but as an example we will show how it can be used to detect changes in mean.

B.1 Usage

To detect changes in mean the function to use is

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

where

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

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

pen.value The value of the penalty when using the Manual penalty option. A vector of length 2 (min,max) if using the CROPS penalty.

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

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

test.stat The assumed test statistic/distribution of the data. Currently only "empirical_distribution".

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

minseglen Positive integer giving the minimum segment length (number of observations between changes), default is the minimum allowed by theory. param.estimates

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

nquantiles The number of quantiles to calculate when test.stat = "empirical_distribution".

To use CROPS we need to set method = "PELT", penalty = "CROPS" and pen.value is a vector of length 2.

B.2 Example

As an example we can simulate data from a Gaussian distribution with a change in mean and use CROPS to detect changes for a range of penalties.

```
set.seed(1)
x=c(rnorm(50,0,1),rnorm(50,5,1),rnorm(50,10,1),rnorm(50,3,1))
out=cpt.mean(x, pen.value = c(4,1500),penalty = "CROPS",method = "PELT")
```

The output is give as an S4 class where we can access the segmentations for a range of penalties by

```
out@cpts.full
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   50   96  100  133  150  159  180
[2,]   50   96  100  133  150   NA   NA
[3,]   50  100  133  150   NA   NA   NA
[4,]   50  100  150   NA   NA   NA   NA
[5,]   50  150   NA   NA   NA   NA   NA
[6,]   50   NA   NA   NA   NA   NA   NA
[7,]   NA   NA   NA   NA   NA   NA   NA
```

Appendix C

Supplementary Material for Chapter 4

Below are further segmentations of the heart-rate data in Chapter 4 using the ED-PELT algorithm and also PELT with a change in slope cost function.

C.1 Further Results - ED-PELT

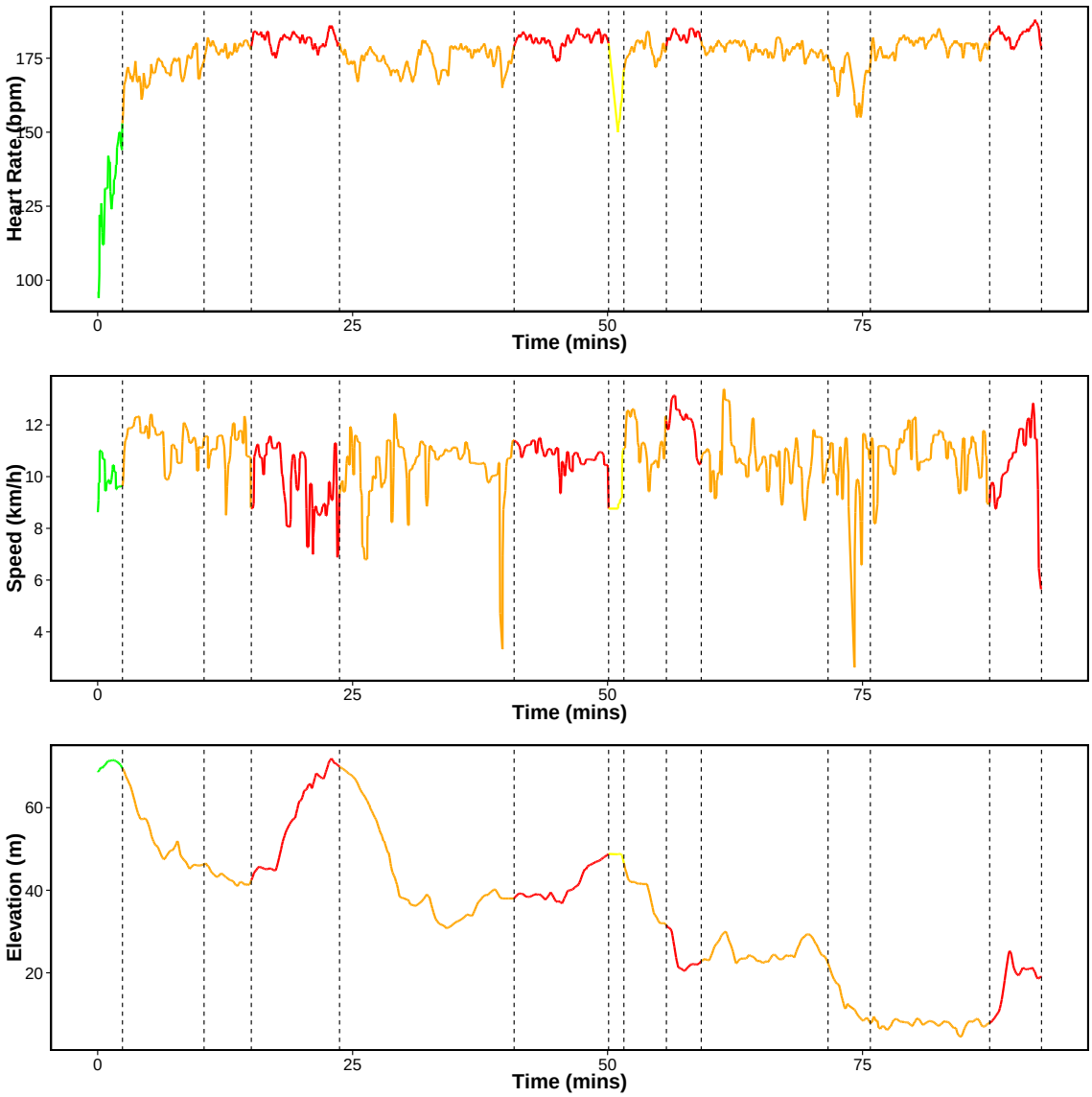


Figure C.1: Segmentations using ED-PELT with 13 changepoints. We have colour coded the line based on the average heart-rate of each segment where red: peak, orange: anaerobic, yellow: aerobic and green: recovery.

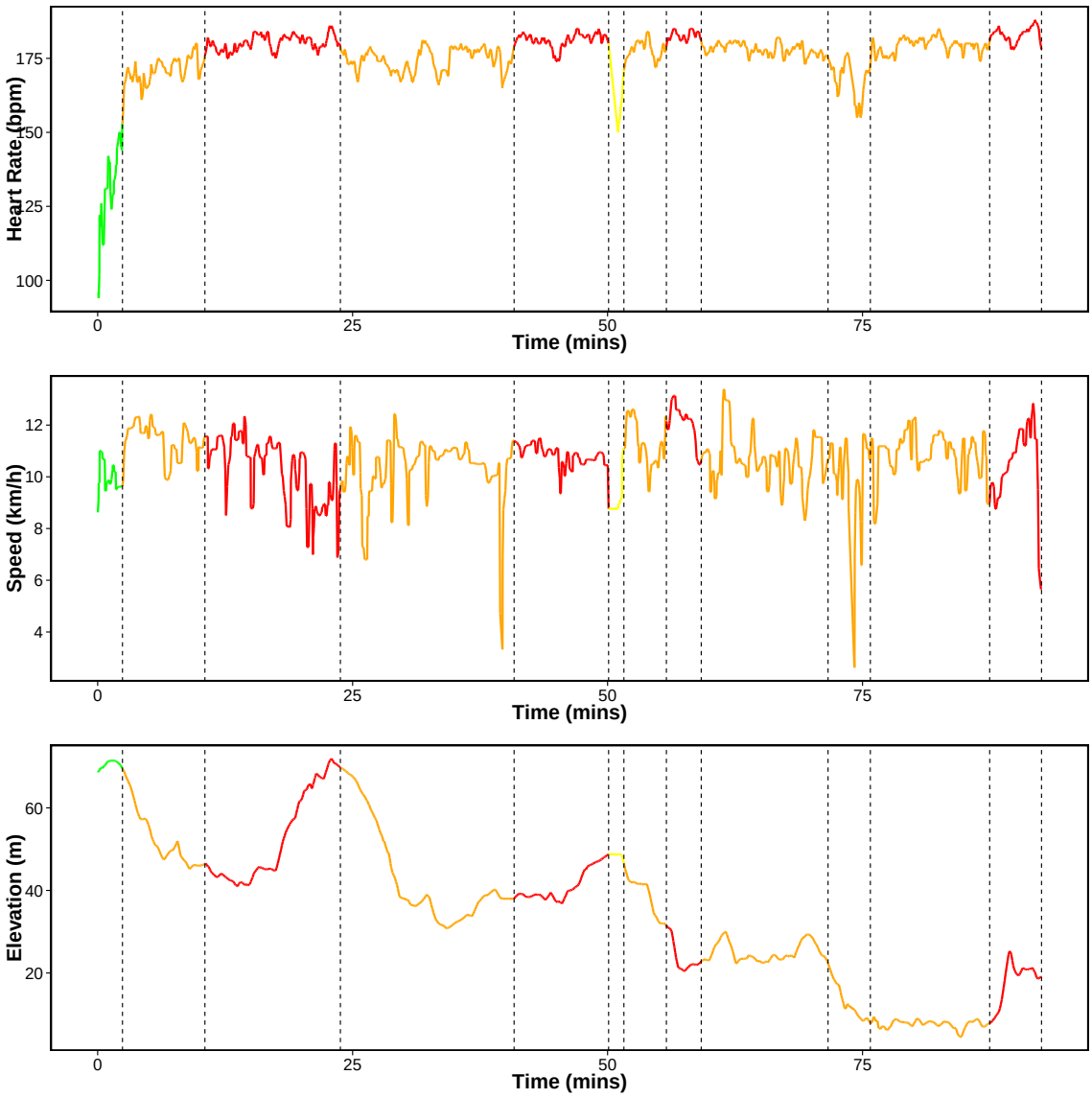


Figure C.2: Segmentations using ED-PELT with 12 changepoints. We have colour coded the line based on the average heart-rate of each segment where red: peak, orange: anaerobic, yellow: aerobic and green: recovery.

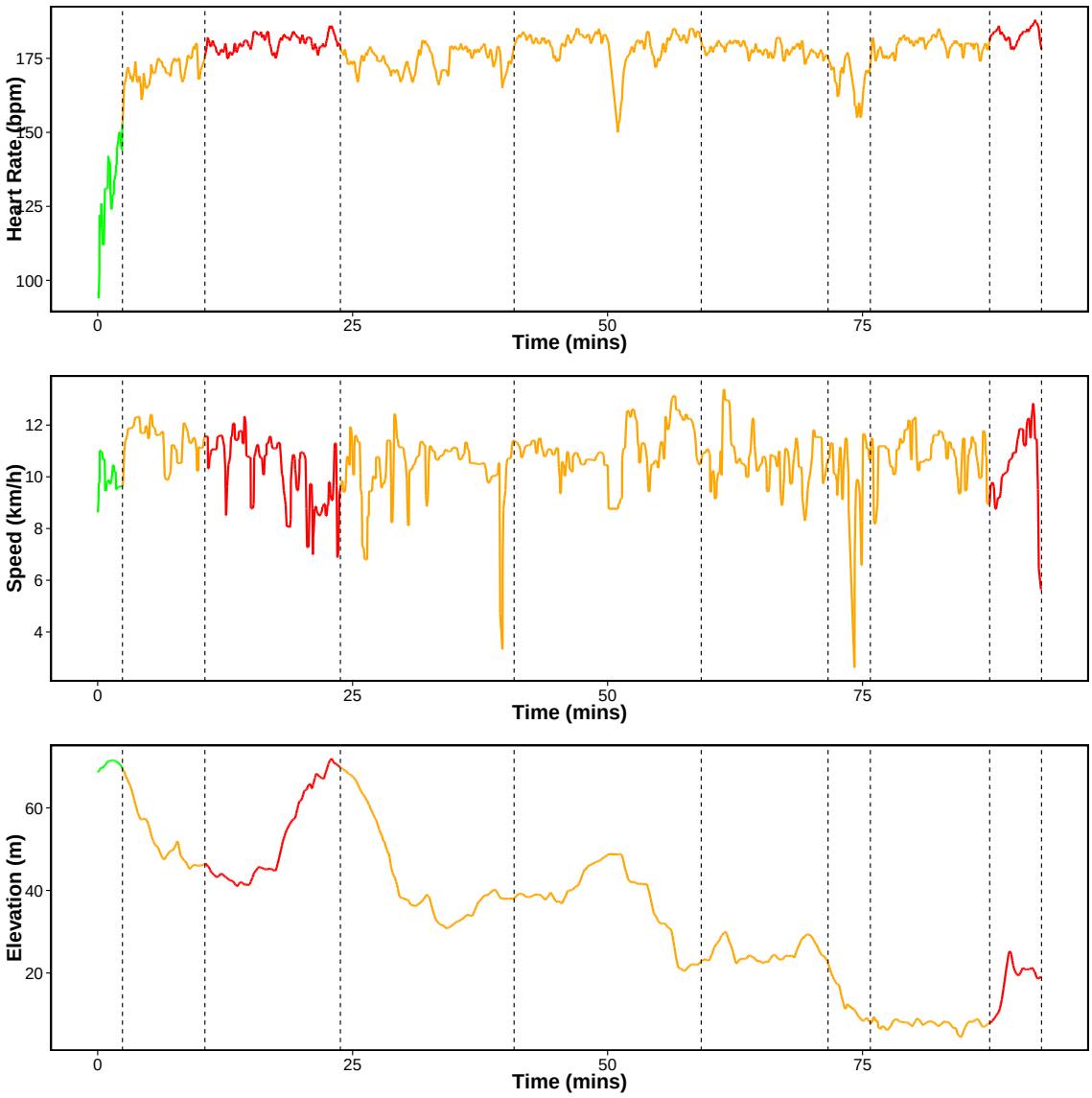


Figure C.3: Segmentations using ED-PELT with 9 changepoints. We have colour coded the line based on the average heart-rate of each segment where red: peak, orange: anaerobic, yellow: aerobic and green: recovery.

C.2 Further Results - Piece-wise linear

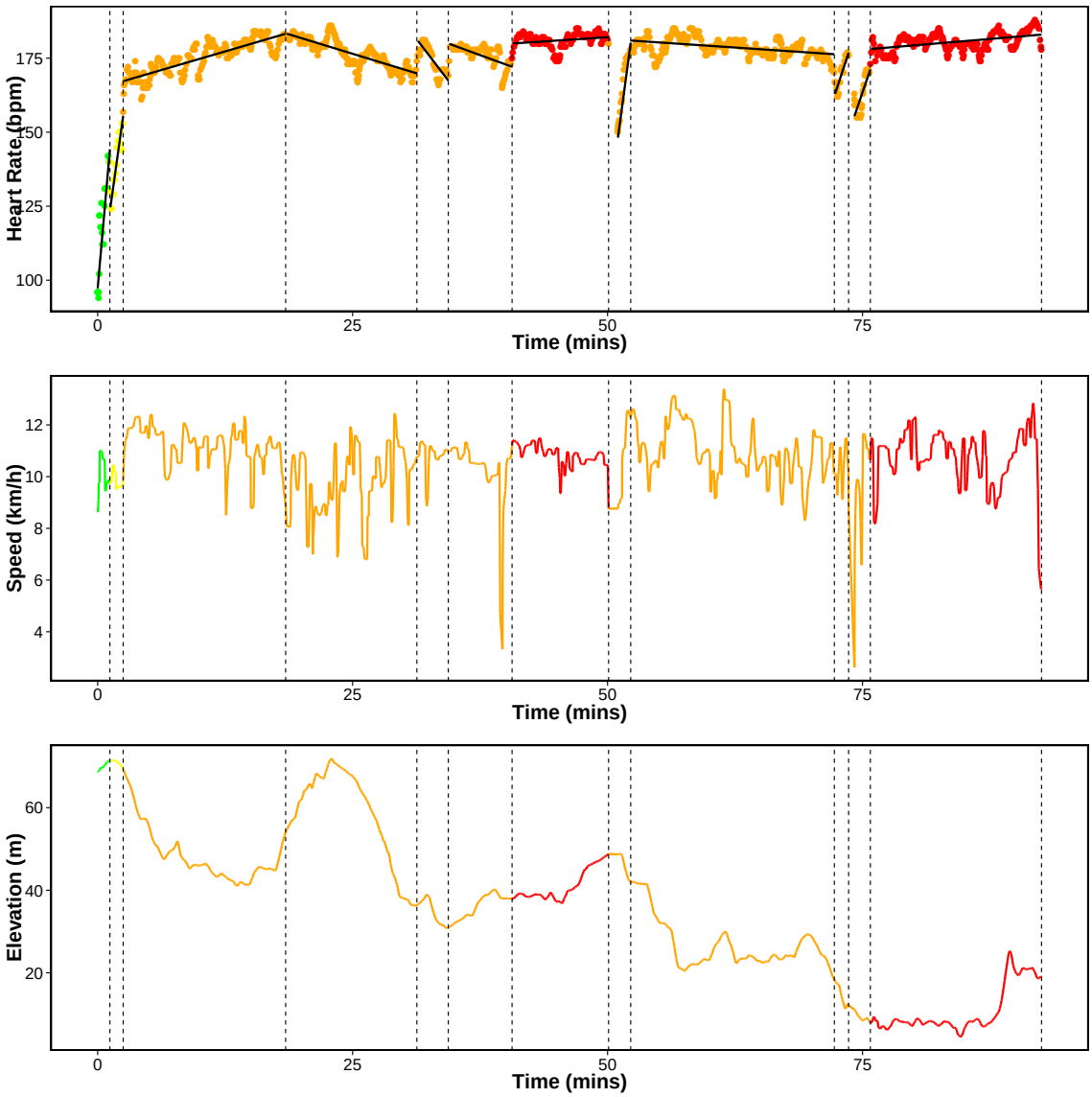


Figure C.4: Segmentations using change in slope with 12 changepoints. We have colour coded the line based on the average heart-rate of each segment where red: peak, orange: anaerobic, yellow: aerobic and green: recovery. The solid black line in the top plot is the best fit for the mean within each segment.

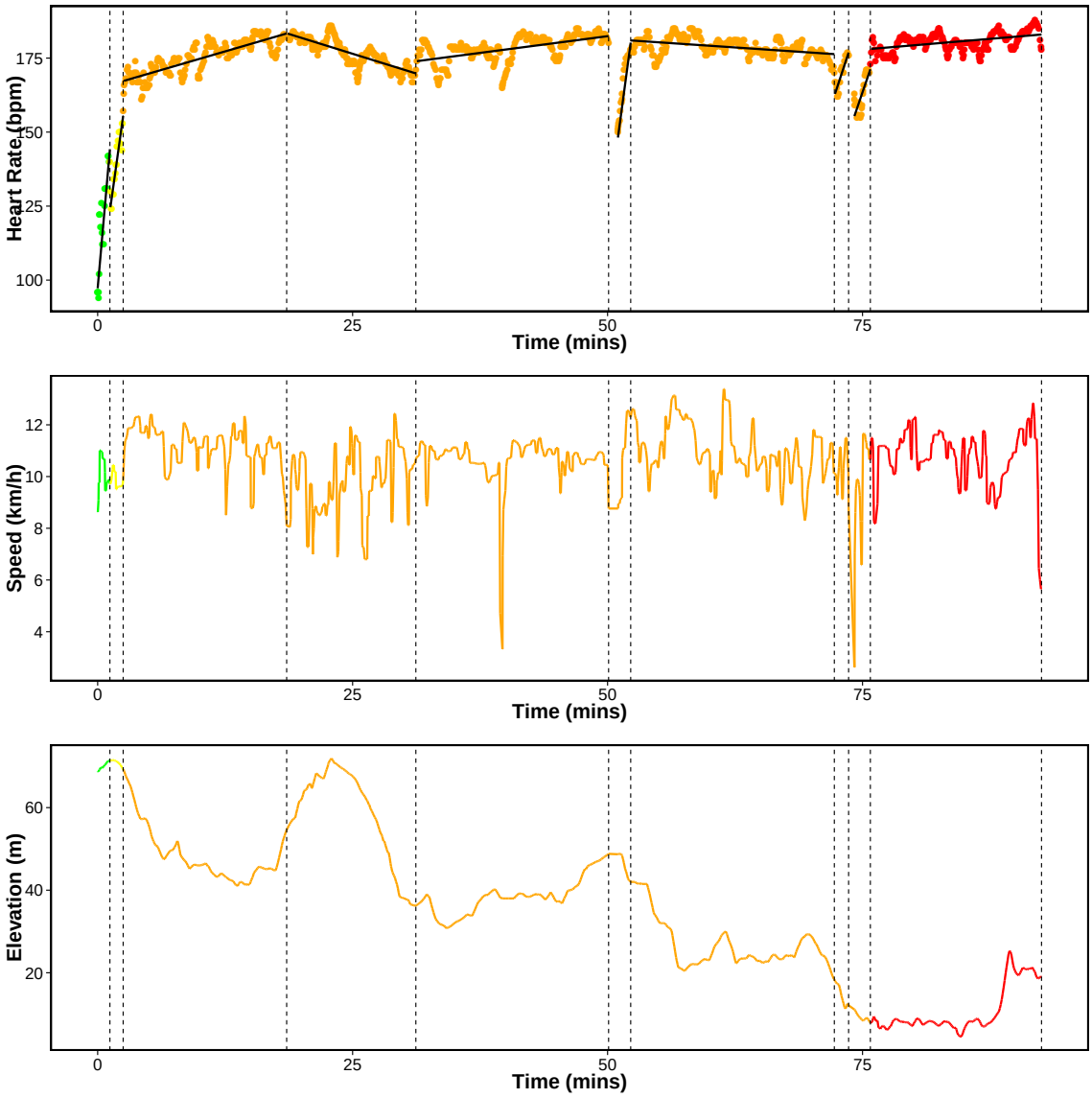


Figure C.5: Segmentations using change in slope with 10 changepoints. We have colour coded the line based on the average heart-rate of each segment where red: peak, orange: anaerobic, yellow: aerobic and green: recovery. The solid black line in the top plot is the best fit for the mean within each segment.

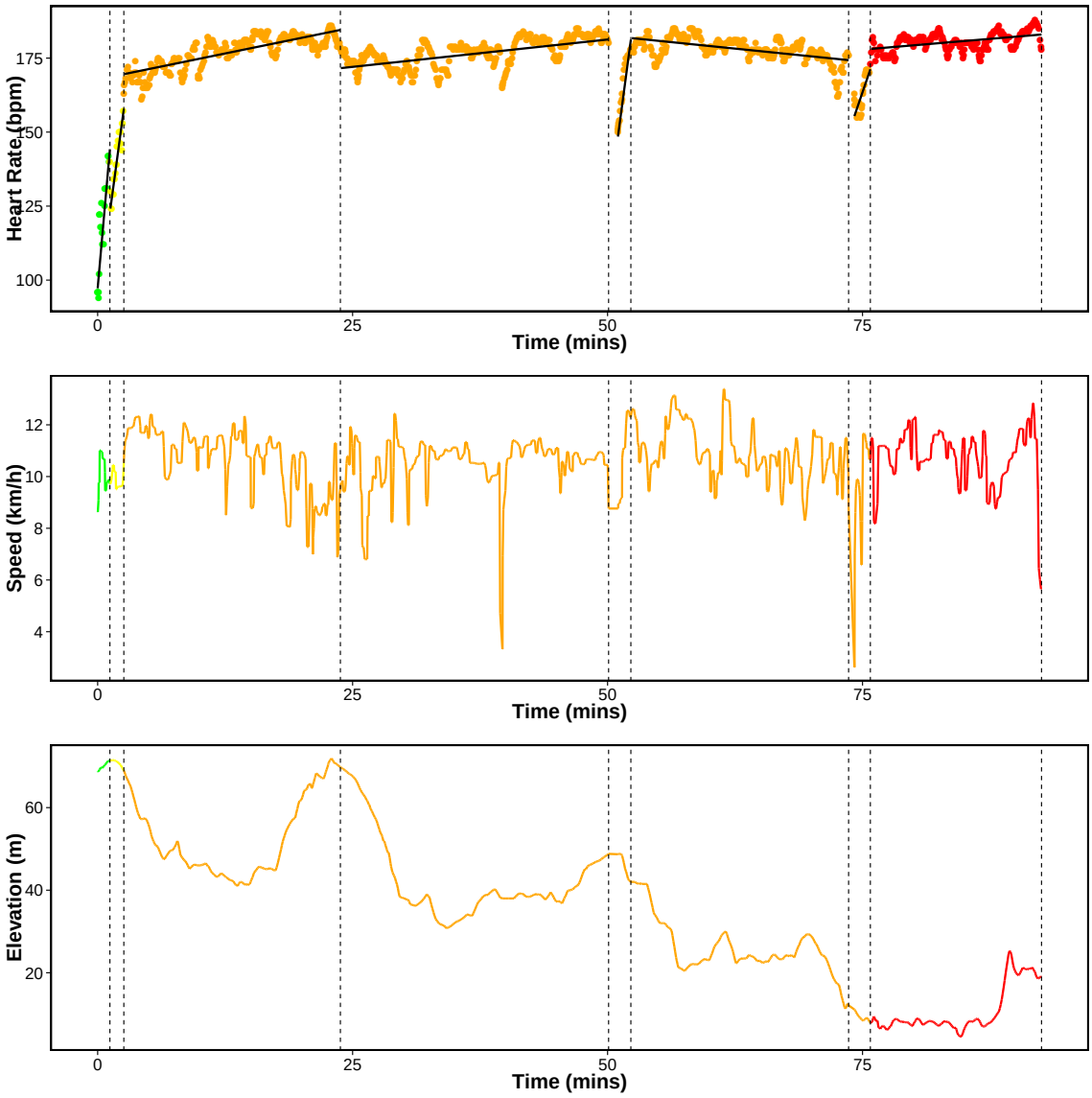


Figure C.6: Segmentations using change in slope with 8 changepoints. We have colour coded the line based on the average heart-rate of each segment where red: peak, orange: anaerobic, yellow: aerobic and green: recovery. The solid black line in the top plot is the best fit for the mean within each segment.

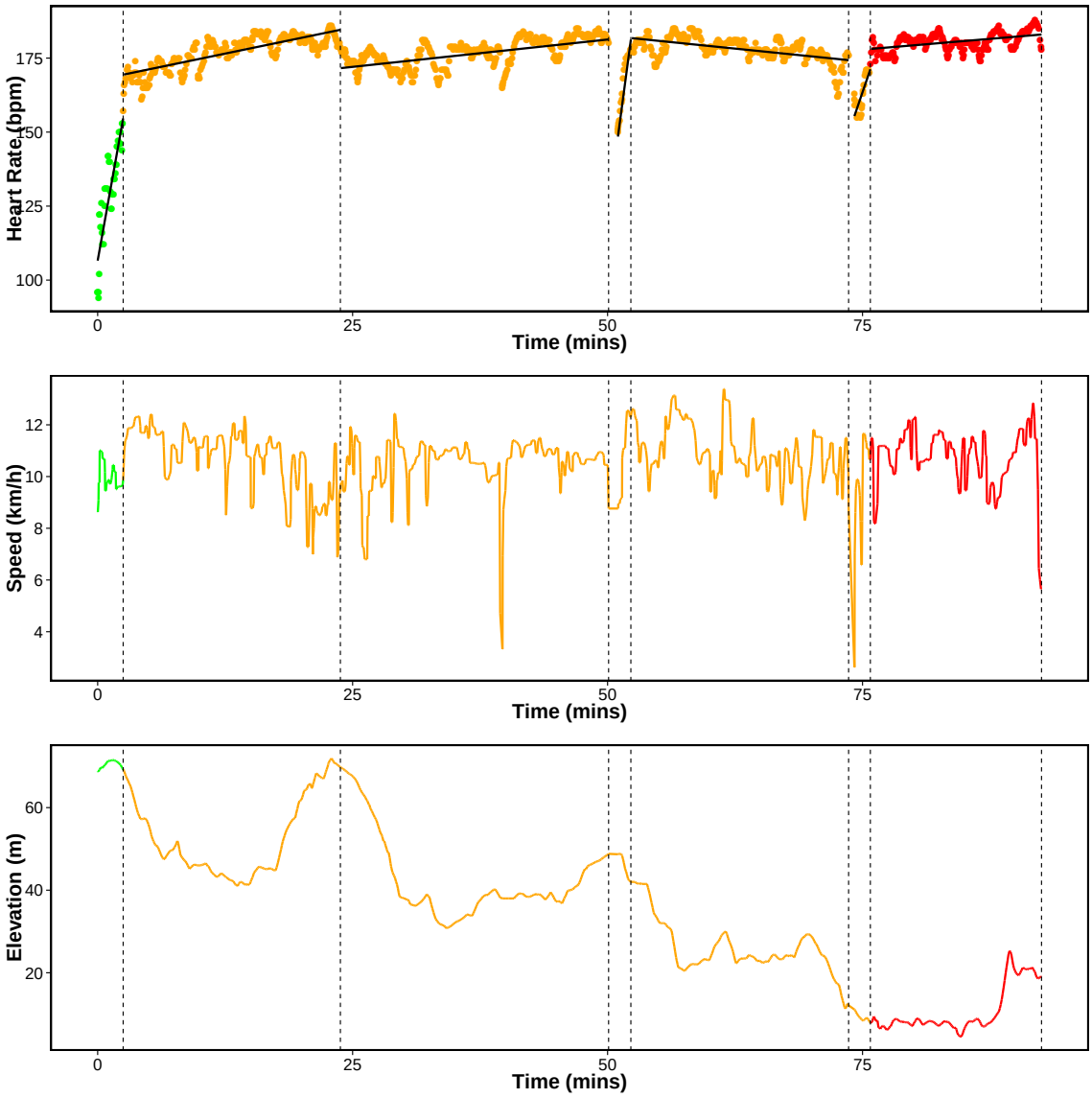


Figure C.7: Segmentations using change in slope with 7 changepoints. We have colour coded the line based on the average heart-rate of each segment where red: peak, orange: anaerobic, yellow: aerobic and green: recovery. The solid black line in the top plot is the best fit for the mean within each segment.

Appendix D

`changepoint.np`: An R Package for Nonparametric Changepoint Detection

The code for the ED-PELT function proposed in Chapter 4 of this thesis can be found in the `changepoint.np` R package. This package is an extension to the `changepoint` package and in fact shares many of the common functions and class objects with this package. Currently ED-PELT is the only function in this package but there is scope to add other methods including Binary Segmentation with a nonparametric cost function such as the cumulative sums of squares. Currently this is part of the `changepoint` package but there is a plan to make this package specifically for parametric changepoint detection and move any nonparametric functions into `changepoint.np`. This package can be downloaded from
https://cran.r-project.org/web/packages/changepoint.np/index.html or from
https://github.com/KayleaHaynes/changepoint.np.

D.1 Package Structure

The main function of this package is `cpt.np` which is structured as follows, analogous to the `cpt.mean`, `cpt.var` and `cpt.meanvar` functions of the `changepoint` package.

D.1.1 Inputs

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

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

pen.value The value of the penalty when using the Manual penalty option. A vector of length 2 (min,max) if using the CROPS penalty.

method Currently the only method is "PELT".

test.stat The assumed test statistic/distribution of the data. Currently only "empirical_distribution".

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

minseglen Positive integer giving the minimum segment length (number of observations between changes), default is the minimum allowed by theory.

nquantiles The number of quantiles to calculate when test.stat = "empirical_distribution".

D.1.2 Outputs

If class = TRUE then an object of S4 class "cpt" is returned. The "cpt" class contains the following slots: data.set, cpttype, method, test.stat, pen.type, pen.value, minseglen, cpts, ncpts.max, param.est. These slots can be accessed using the @ symbol.

If class = FALSE then the structure is as follows. If the method is PELT then a vector is returned containing the changepoint locations for the penalty supplied. If the penalty is CROPS then a list is returned with the elements:

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

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

If the data input is a vector then the corresponding vector or list is returned. If the data is a matrix then a list is returned where each element of the list is either a vector or a list.

D.2 Examples

We now show a couple of examples on how the functions can be used.

D.2.1 Simulated Data

Firstly we simulate data from model 1 of Chapter 4. That is:

```
set.seed(12)


J <- function(x){
```

```
   (1+sign(x))/2
 }


n <- 1000
tau <- c(0.1,0.13,0.15,0.23,0.25,0.4,0.44,0.65,0.76,0.78,0.81)*n
h <- c(2.01, -2.51, 1.51, -2.01, 2.51, -2.11, 1.05, 2.16, -1.56, 2.56, -2.11)
sigma <- 0.5
t <- seq(0,1,length.out = n)
data <- array()
for (i in 1:n){
   data[i] <- sum(h*J(n*t[i] - tau)) + (sigma * rnorm(1))
}
```

we can use ED-PELT to find the changepoints using the SIC penalty value.

```
out <- cpt.np(data, penalty = "SIC",method="PELT",
test.stat="empirical_distribution",
             class=TRUE,minseglen=2, nquantiles =4*log(length(data)))
```

The outputted changepoints can be detected using either `cpts(out)` or `out@cpts`. This returns

```
cpts(out)
 [1] 100 130 150 230 250 400 440 650 760 780 810.
```

In order to visualise the changepoints there is a plot method for the cpt class. That is `plot(out)` will return the plot in Figure D.1.

D.2.2 Heart-Rate Data

The second example we look at is the Heart-Rate data recorded during a period of activity as in Chapter 4. This data can be found and used in the `changepoint.np` package. This time we will show an example of using the CROPS penalty term.

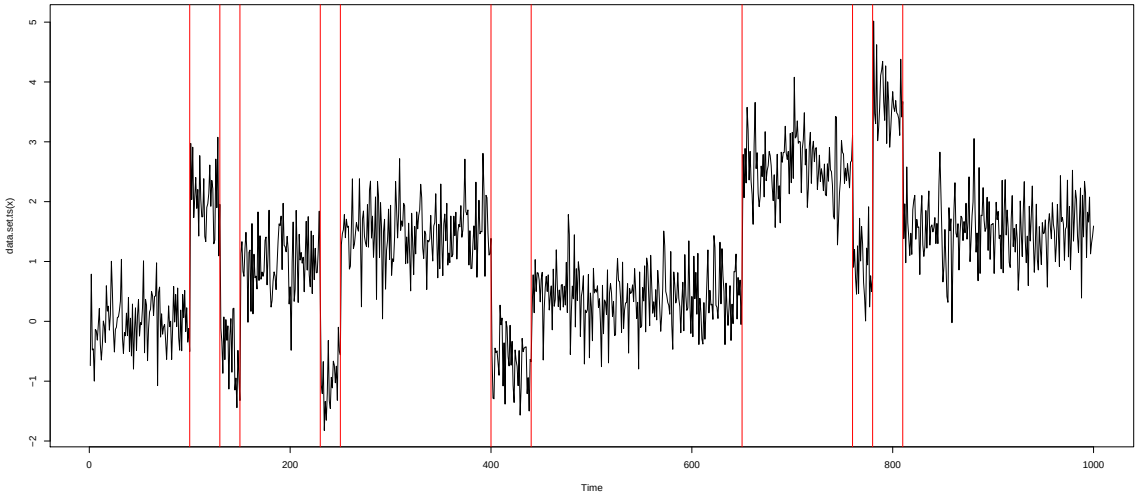


Figure D.1: Plot of the changepoints using the plot method of the cpt class

```
cptHeartRate <- cpt.np(HeartRate, penalty = "CROPS", pen.value = c(10,100),
 method="PELT", test.stat="empirical_distribution",class=TRUE,minseglen=2,
  nquantiles =4*log(length(HeartRate)))
Class 'cpt' : Changepoint Object
       ~~   : S4 class containing 14 slots with names
              cpts.full pen.value.full data.set cpttype method test.stat
               pen.type pen.value minseglen cpts ncpts.max param.est
               date version
Created on  : Tue Aug  2 17:15:27 2016
summary(.)  :
----------
Created Using changepoint version 2.2.1
Changepoint type      : Change in nonparametric (empirical_distribution)
Method of analysis    : PELT
Test Statistic  : empirical_distribution
Type of penalty       : CROPS with value, 10 100
Minimum Segment Length : 2
```

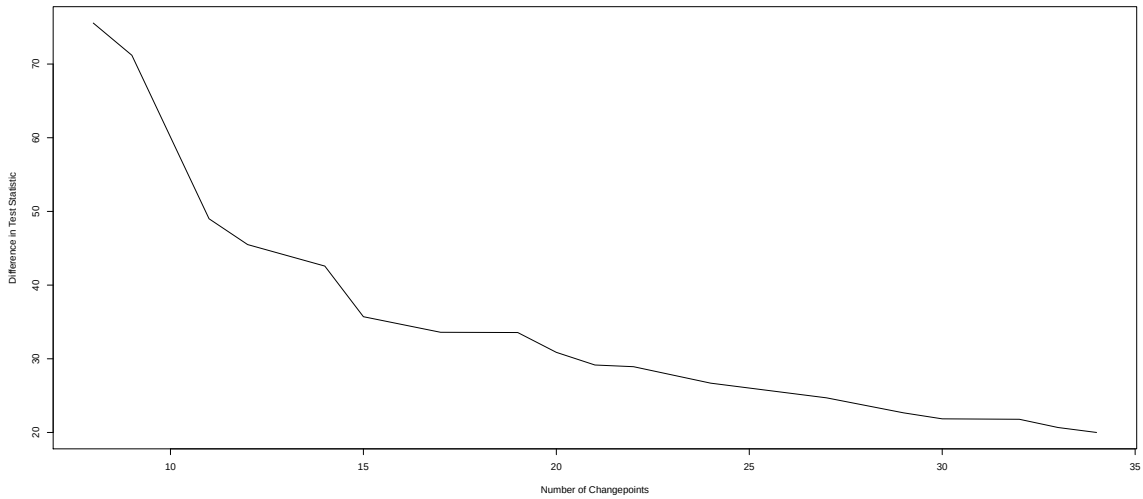


Figure D.2: Diagnostic plot for the heart-rate data-set when the CROPS penalty is used.

```
Maximum no. of cpts   :
Changepoint Locations :
Number of segmentations recorded: 29  with between  8  and  50
changepoints.
 Penalty value ranges from: 10  to  75.56285
```

This time when we plot we have to decide how many changepoints to include. If unsure a diagnostic plot can be used where an elbow is plotted and the "best" segmentations are the ones around the elbow. A similar method is discussed in Chapters 3 and 4.

```
plot(cptHeartRate, diagnostic = TRUE)
plot(cptHeartRate, ncpts = 15)
```

For this example the diagnostic plot is shown in Figure D.2 and the segmentation with 15 changepoints in Figure D.3.

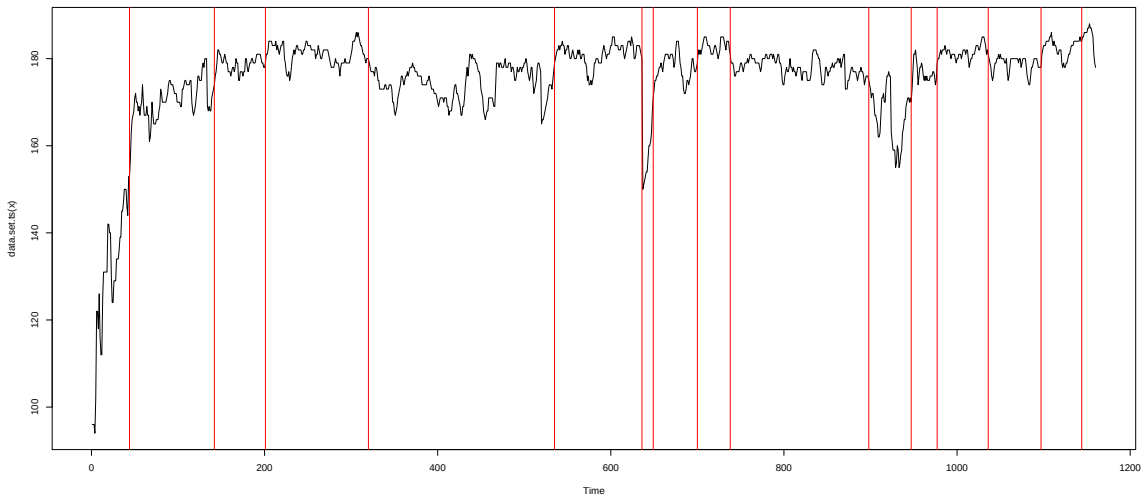


Figure D.3: Segmentation of the heart-rate data with 15 changepoints.

Appendix E

Further Simulation Results for Chapter 5

In Section 5.4.3 we looked at the performance of SM1 and SM2 on the teeth and stairs signals with different data lengths and size of segments. In the main body of text we showed the results for segment lengths equal to 5, 20 and 100. In Figure E.1 and Figure E.2 we show further results for segment lengths equal to 10, 15 and 50. In the teeth signal SM2 performs better than SM1 when the length of the segments and size of the changes are large. When the segment lengths are small SM1 performs better but the accuracy decays as the number of cores increases. In the stairs signal SM2 performs better than SM1 in all cases. These results follow from the results shown in the main body of text.

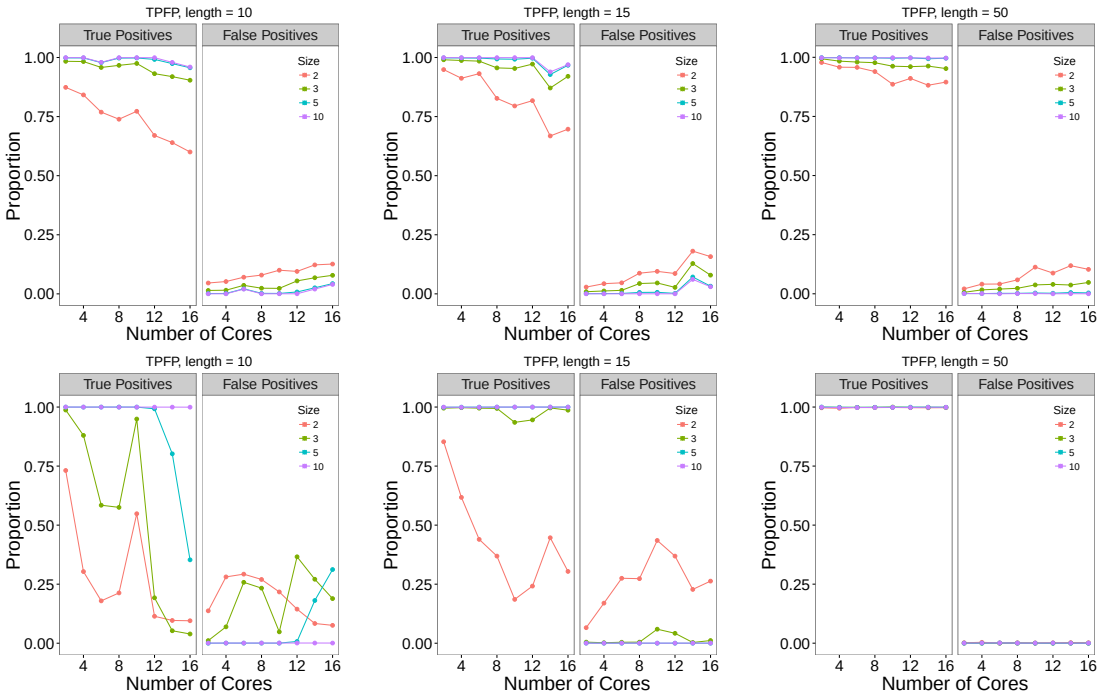


Figure E.1: The TFDR for SM1 (top) and SM2 (bottom) on the teeth signal over a different number of cores compared to detecting the changes on one core. The plots left to right are different lengths of the segments: left - 10, centre - 15 and right - 50. The different coloured lines on each plot are different jump sizes.

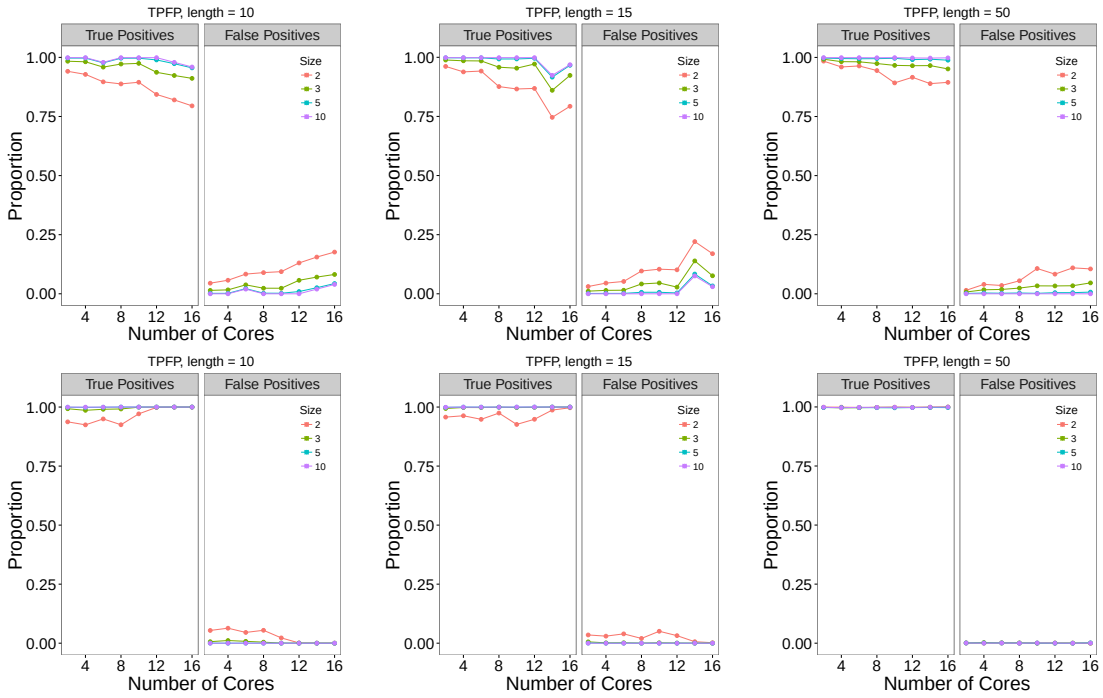


Figure E.2: The TFDR for SM1 (top) and SM2 (bottom) on the stairs signal over a different number of cores compared to detecting the changes on one core. The plots left to right are different lengths of the segments: left - 10, centre - 15 and right - 50. The different coloured lines on each plot are different jump sizes.

Appendix F

R Code for the SM1 and SM2 methods proposed in Chapter 5

The code for the parallel algorithms proposed in Chapter 5 of this thesis can be found in the `changepoint.parallel` R package that can be downloaded from Github https://github.com/KayleaHaynes/changepoint.parallel. This package currently only runs SM1 and SM2 with PELT.

F.1 Package Structure

The two main functions in this package are `Parallel_PELTSM1` and `Parallel_PELTSM2`.

Common Inputs

The below inputs are required for both SM1 and SM2.

data A vector of data-points within which you with to find changepoints.

penalty The value of the penalty.

pruning If true PELT is used, if false Optimal Partitioning is used.

sum.stat This can be "norm.sum" or "exp.sum".

cost This can be "norm.mean.cost", "norm.var.cost", "norm.meanvar.cost" or "exp.cost".

ncores Number of cores to use.

minseglen Minimum length a segment can be.

Additional Inputs for SM1

Below are additional inputs used for SM1, these describe the boundary that should be used.

boundary This can either be "fixed" or "adaptive".

boundary_value If boundary is fixed then this is the number of points to use around the boundary.

F.1.1 Output

The output of both functions is a vector of the changepoint locations.

F.1.2 Example

Below is an example of SM1 and SM2 on the blocks data-set.

```
### Set up parallel environment ###
library(doParallel)
library(foreach)


ncores <- c(10)
cl <- makeCluster(ncores)
registerDoParallel(cl)


### Generate some data from the blocks data-set ####
cpts <- round(c(0,0.1, 0.13, 0.15, 0.23, 0.25, 0.40, 0.44, 0.65, 0.76,
```

```
 0.78, 0.81,1)*10000)
segment_param <- c(0, 4, -1, 2, -2, 3, -1.2, 0.9, 5.2, 2.1, 4.2, 0)
data <- NULL

for (i in 1:(length(cpts)-1)){
   data_new <- rep(segment_param[i],cpts[i+1] - cpts[i])
   data <- c(data, data_new)
}
data <- data + rnorm(10000,0,1)

Parallel_PELTSM1(data, 2*log(length(data)), TRUE, sum.stat = norm.sum,
 cost = norm.mean.cost, ncores=10, boundary = "fixed",boundary_value = 20, 1)

[1]     0  1000  1300  1500  2300  2500  4000  4401  6500  7600  7799
 8100 10000

Parallel_PELTSM2(data, 2*log(length(data)), TRUE, sum.stat = norm.sum,
 cost = norm.mean.cost, minseglen =1, ncores=10)

[1]     0  1000  1300  1500  2300  2500  4000  4401  6500  7600  7799
  8100 10000
```

Bibliography

Adams, R. P. and MacKay, D. J. C. (2007). Bayesian Online Changepoint Detection. *arXiv e-prints 0710.3742*.

Aggarwal, R., Inclan, C., and Leal, R. (1999). Volatility in emerging stock markets. *The Journal of Financial and Quantitative Analysis*, 34(1):33–55.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA. ACM.

Andreou, E. and Ghysels, E. (2009). Structural breaks in financial time series. In *Handbook of Financial Time Series*, pages 839–870. Springer Berlin Heidelberg, Berlin, Heidelberg.

Android (2016). Sensors overview. `https://developer.android.com/guide/topics/sensors/sensors_overview.html`. Accessed: 2016-08-01.

Apley, D. W. and Chang-Ho, C. (2007). An optimal filter design approach to statistical process control. *Journal of Quality Technology*, 39(2):93–117.

Aston, J. A. D. and Kirch, C. (2012). Evaluating stationarity via change-point detection alternatives with applications to fMRI data. *The Annals of Applied Statistics*, 6(4):1906–1948.

Aubert, A. E., Seps, B., and Beckers, F. (2003). Heart rate variability in athletes. *Sports Medicine*, 33(12):889–919.

Aue, A., Härmann, S., Horváth, L., and Reimherr, M. (2009). Break detection in the covariance structure of multivariate time series models. *The Annals of Statistics*, 37(6B):4046–4087.

Aue, A. and Horváth, L. (2013). Structural breaks in time series. *Journal of Time Series Analysis*, 34(1):1–16.

Auger, I. E. and Lawrence, C. E. (1989). Algorithms for the optimal identification of segment neighborhoods. *Bulletin of Mathematical Biology*, 51(1):39–54.

Bai, J. and Perron, P. (1998). Estimating and testing linear models with multiple structural changes. *Econometrica*, 66(1):47–78.

Bardwell, L. and Fearnhead, P. (2017). Bayesian detection of abnormal segments in multiple time series. *Bayesian Analysis*, 12(1):193–.

Barney, B. (2016a). Introduction to parallel computing. `https://computing.llnl.gov/tutorials/parallel_comp/`. Accessed: 2016-08-13.

Barney, B. (2016b). Message passing interface (MPI). `https://computing.llnl.gov/tutorials/mpi/`. Accessed: 2016-08-13.

Baron, M. (2000). Nonparametric adaptive change-point estimation and on-line detection. *Sequential Analysis*, 19(1-2):1–23.

Barry, D. and Hartigan, J. A. (1992). Product partition models for change point problems. *The Annals of Statistics*, 20(1):260–279.

Basseville, M. and Nikiforov, I. V. (1993). *Detection of Abrupt Changes: Theory and Application*. Prentice Hall, Englewood Cliffs.

Batsidis, A., Horváth, L., Martín, N., Pardo, L., and Zografos, K. (2013). Change-point detection in multinomial data using phi-divergence test statistics. *Journal of Multivariate Analysis*, 118(1):53 – 66.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA.

Bhattacharyya, G. and Johnson, R. (1968). Nonparametric tests for shift at an unknown time point. *The Annals of Mathematical Statistics*, 39(5):1731–1743.

Billat, V. L., Mille-Hamard, L., Meyer, Y., and Wesfreid, E. (2009). Detection of changes in the fractal scaling of heart rate and speed in a marathon race. *Physica A: Statistical Mechanics and its Applications*, 388(18):3798 – 3808.

Bodenham, D. A. and Adams, N. M. (2013). Continuous monitoring of a computer network using multivariate adaptive estimation. In *2013 IEEE 13th International Conference on Data Mining Workshops*, pages 311–318.

Bodenham, D. A. and Adams, N. M. (2016). Continuous monitoring for changepoints in data streams using adaptive estimation. *Statistics and Computing (To Appear)*, pages 1–14.

Boysen, L., Kempe, A., Liebscher, V., Munk, A., and Wittich, O. (2009). Consistencies and rates of convergence of jump-penalized least squares estimators. *The Annals of Statistics*, 37(1):157–183.

BrainMacSportsCoach (2015). Heart Rate Training Zones. https://http://www.brianmac.co.uk/hrm1.htm.

Brault, V., Delattre, M., Lebarbier, E., Mary-Huard, T., and Lévy-Leduc, C. (2015). Estimating the number of change-points in a two-dimensional segmentation model without penalization. *arXiv e-prints 1506.03198*.

Braun, J., R.K., M., and Mueller, H.-G. (2000). Multiple changepoint fitting via quasilikelihood, with application to DNA sequence segmentation. *Biometrika*, 87(2):301–314.

Braun, J. V. and Müeller, H.-G. (1998). Statistical methods for DNA sequence segmentation. *Statistical Science*, 13(2):142–162.

Brodsky, B. E. and Darkhovsky, B. S. (1993). *Nonparametric methods in change-point problems*, volume 243 of *Mathematics and its Applications*. Kluwer Academic Publishers Group, Dordrecht.

Butenhof, D. R. (1997). *Programming with POSIX Threads*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Calaway, R., RevolutionAnalytics, and Weston, S. (2015). *foreach: Provides Foreach Looping Construct for R*. R package version 1.4.3.

Calaway, R., RevolutionAnalytics, Weston, S., and Tenenbaum, D. (2014). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.10.

Capizzi, G. and Masarotto, G. (2012). Adaptive generalized likelihood ratio control charts for detecting unknown patterned mean shifts. *Journal of Quality Technology*, 44(4):281–303.

Cappé, O., Moulines, E., and Ryden, T. (2005). *Inference in Hidden Markov Models (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Carlstein, E. (1988). Nonparametric change-point estimation. *The Annals of Statistics*, 16(1):188–197.

Chakraborti, S. and van de Wiel, M. A. (2008). *A nonparametric control chart based on the Mann-Whitney statistic*, volume 1 of *Collections*, pages 156–172. Institute of Mathematical Statistics, Beachwood, Ohio, USA.

Chen, J. and Gupta, A. K. (1997). Testing and locating variance changepoints with application to stock prices. *Journal of the American Statistical Association*, 92(438):739–747.

Chen, J. and Gupta, A. K. (2000). *Parametric statistical change point analysis*. Birkhäuser Boston Inc., Boston, MA.

Chib, S. (1996). Calculating posterior distributions and modal estimates in markov mixture models. *Journal of Econometrics*, 75(1):79 – 97.

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

Cho, H. and Fryzlewicz, P. (2015). Multiple-change-point detection for high dimensional time series via sparsified binary segmentation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(2):475–507.

Cleynen, A., Dudoit, S., and Robin, S. (2014). Comparing segmentation methods for genome annotation based on RNA-seq data. *Journal of Agricultural, Biological, and Environmental Statistics*, 19(1):101–118.

Cleynen, A., Koskas, M., Lebarbier, E., Rigaill, G., and Robin, S. (2013). Segmentor3IsBack: an R package for the fast and exact segmentation of Seq-data. *arXiv eprints 1204.5564*.

Csörgö, M. and Horváth, L. (1997). *Limit Theorems in Change-point Analysis*. John Wiley & Sons Ltd., Chichester.

Dagum, L. and Menon, R. (1998). OpenMP: an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering*, 5(1):46–55.

Davis, R. A., Lee, T. C. M., and Rodriguez-Yam, G. A. (2006). Structural break estimation for nonstationary time series models. *Journal of the American Statistical Association*, 101(473):223–239.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 39(1):1–38.

Dette, H. and Wied, D. (2016). Detecting relevant changes in time series models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(2):371–394.

Donoho, D. L. and Johnstone, I. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455.

Dümbgen, L. (1991). The asymptotic behavior of some nonparametric change-point estimators. *The Annals of Statistics*, 19(3):1471–1495.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press.

Eckley, I. A., Fearnhead, P., and Killick, R. (2011). Analysis of changepoint models. In *Bayesian Time Series Models*. Cambridge University Press.

Eddelbuettel, D. (2016). CRAN task view: High-performance and parallel computing with r. `https://cran.r-project.org/web/views/HighPerformanceComputing.html`. Accessed: 2016-08-13.

Fan, J. and Wang, Y. (2007). Multi-scale jump and volatility analysis for high-frequency financial data. *Journal of the American Statistical Association*, 102(480):1349–1362.

Fan, T.-H., Lin, D. K., and Cheng, K.-F. (2007). Regression analysis for massive datasets. *Data & Knowledge Engineering*, 61(3):554 – 562. Advances on Natural Language Processing - NLDB 05.

Fearnhead, P. (2005). Exact bayesian curve fitting and signal segmentation. *IEEE Transactions on Signal Processing*, 53(6):2160–2166.

Fearnhead, P. (2006). Exact and efficient Bayesian inference for multiple changepoint problems. *Statistics and Computing*, 16(2):203–213.

Fearnhead, P. and Clifford, P. (2003). On-line inference for hidden markov models via particle filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(4):887–899.

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

Fearnhead, P. and Liu, Z. (2011). Efficient Bayesian analysis of multiple changepoint models with dependence across segments. *Statistics and Computing*, 21(2):217–229.

Fernandez, V. (2004). Detection of breakpoints in volatility. Documentos de Trabajo 194, Centro de Economa Aplicada, Universidad de Chile.

Frick, K., Munk, A., and Sieling, H. (2014). Multiscale change-point inference. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(3):495–580.

Fryzlewicz, P. (2012). Timethreshold maps: Using information from wavelet reconstructions with all threshold values simultaneously. *Journal of the Korean Statistical Society*, 41(2):145 – 159.

Fryzlewicz, P. (2014). Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6):2243–2281.

Futschik, A., Hotz, T., Munk, A., and Sieling, H. (2014). Multiresolution DNA partitioning: statistical evidence for segments. *Bioinformatics*, 30(16):2255–2262.

Galway, L., Zhang, S., Nugent, C., McClean, S., Finlay, D., and Scotney, B. (2011). Utilizing wearable sensors to investigate the impact of everyday activities on heart rate. In Abdulrazak, B., Giroux, S., Bouchard, B., Pigot, H., and Mokhtari, M., editors, *Toward Useful Services for Elderly and People with Disabilities*, volume 6719 of *Lecture Notes in Computer Science*, pages 184–191. Springer Berlin Heidelberg.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, B., and Sunderam, V. (1994). *PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambrige.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732.

Guan, Z. (2004). A semiparametric changepoint model. *Biometrika*, 91(4):849–862.

Guarnaccia, C., Quartieri, J., Tepedino, C., and Rodrigues, E. R. (2015). An analysis of airport noise data using a non-homogeneous Poisson model with a change-point. *Applied Acoustics*, 91(1):33–39.

Haccou, P., Meelis, E., and van de Geer, S. (1987). The likelihood ratio test for the change point problem for exponentially distributed random variables. *Stochastic Processes and their Applications*, 27(1):121–139.

Hand, D. J. and Weston, D. J. (2008). *Statistical Techniques for Fraud Detection, Prevention and Assessment*, pages 257–270. IOS PRess.

Hannan, E. and Quinn, B. (1979). The determination of the order of an autoregression. *Journal of the Royal Statistical Society: Series B*, 41(2):190–195.

Hawkins, D. M. and Deng, Q. (2010). A Nonparametric Change-Point Control Chart. *Journal of Quality Technology*, 42(2):165–173.

Hawkins, D. M., Qui, P., and Kang, C. (2003). The changepoint model for statistical process control. *Journal of Quality Technology*, 35(4):355–366.

Hawkins, D. M. and Zamba, K. D. (2005). A change-point model for a shift in variance. *Journal of Quality Technology*, 37(1):21–31.

Haynes, K. (2016). *changepoint.np: Methods for Nonparametric Changepoint Detection*. R package version 0.0.2.

Haynes, K., Eckley, I. A., and Fearnhead, P. (2017). Computationally efficient changepoint detection for a range of penalties. *Journal of Computational and Graphical Statistics*, 26(1):134–143.

Haynes, K., Fearnhead, P., and Eckley, I. A. (2016). A computationally efficient nonparametric approach for changepoint detection. *Statistics and Computing (To appear)*.

Heard, N. A., Weston, D. J., Platanioti, K., and Hand, D. J. (2010). Bayesian anomaly detection methods for social networks. *The Annals of Applied Statistics*, 4(2):645–662.

Hegland, M., McIntosh, I., and Turlach, B. A. (1999). A parallel solver for generalised additive models. *Computational Statistics and Data Analysis*, 31(4):377–396.

Hinkley, D. V. (1970). Inference about the change-point in a sequence of random variables. *Biometrika*, 57(1):1–17.

Hinkley, D. V. and Hinkley, E. A. (1970). Inference about the change-point in a sequence of binomial variables. *Biometrika*, 57(3):477–488.

Hocking, T., Rigaill, G., Vert, J.-P., and Bach, F. (2013a). Learning sparse penalties for change-point detection using max margin interval regression. In *ICML (3)*, volume 28 of *JMLR Proceedings*, pages 172–180.

Hocking, T., Schleiermacher, G., Janoueix-Lerosey, I., Boeva, V., Cappo, J., Delattre, O., Bach, F., and Vert, J.-P. (2013b). Learning smoothing models of copy number profiles using breakpoint annotations. *BMC Bioinformatics*, 14(1):1–15.

Horváth, L. and Hušková, M. (2012). Change-point detection in panel data. *Journal of Time Series Analysis*, 33(4):631–648.

Hotz, T., Schütte, O. M., Sieling, H., Polupanow, T., Diederichsen, U., Steinem, C., and Munk, A. (2013). Idealizing ion channel recordings by jump segmentation and statistical multiresolution analysis. *IEEE Transactions of Nanobioscience*, 12(4):376–386.

Inclán, C. and Tiao, G. C. (1994). Use of cumulative sums of squares for retrospective detection of changes of variance. *Journal of the American Statistical Association*, 89(427):913–923.

Jackson, B., Sargle, J., Barnes, D., Arabhi, S., Alt, A., Gioumousis, P., Gwin, E., Sangtrakulcharoen, P., Tan, L., and Tsai, T. (2005). An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2):105–108.

James, N. A. and Matteson, D. S. (2015). Change Points via Probabilistically Pruned Objectives. *arXiv eprints 1505.04302*.

Jandhyala, V., Fotopoulos, S., MacNeill, I., and Liu, P. (2013). Inference for single and multiple change-points in time series. *Journal of Time Series Analysis*, 34(4):423–446.

Jeng, X. J., Cai, T. T., and Li, H. (2013). Simultaneous discovery of rare and common segment variants. *Biometrika*, 100(1):157–172.

Jiang, W., Shu, L., and Apley, D. W. (2008). Adaptive CUSUM procedures with EWMA-based shift estimators. *IIE Transactions*, 40(10):992–1003.

Jones, L. A. (2002). The statistical design of EWMA control charts with estimated parameters. *Journal of Quality Technology*, 34(3):277–288.

Karolos, K. K. and Fryzlewicz, P. (2016). Multiple change-point detection for non-stationary time series using wild binary segmentation. *Statistica Sinica (To Appear)*.

Khalfa, N., Bertandm, P. R., Boudet, G., Chamoux, A., and Billat, V. (2012). Heart rate regulation processed through wavelet analysis and change detection: some case studies. *Acta Biotheoretica*, 60(1):109 – 29.

Killick, R. and Eckley, I. A. (2014). changepoint: An R Package for Changepoint Analysis. *Journal of Statistical Software*, 58(3):1–19.

Killick, R., Eckley, I. A., and Haynes, K. (2014). *changepoint: An R package for changepoint analysis*. R package version 2.1.1.

Killick, R., Eckley, I. A., and Jonathan, P. (2013). A wavelet-based approach for detecting changes in second order structure within nonstationary time series. *Electronic Journal of Statistics*, 7(1):1167–1183.

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. *Journal of the American Statisical Association*, 107(500):1590–1598.

Kim, H., Rozovskii, B. L., and Tartakovsky, A. G. (2004). A nonparametric multichart CUSUM test for rapid detection of DOS attacks in computer networks. *International Journal of Computing & Information Sciences*, 2(3):149–158.

Kirch, C., Muhsal, B., and Ombao, H. (2015). Detection of changes in multivariate time series with application to EEG data. *Journal of the American Statistical Association*, 110(511):1197–1216.

Kulkarni, V., Al-Rfou, R., Perozzi, B., and Skiena, S. (2015). Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635. ACM.

Lai, T. L. (2001). Sequential analysis: Some classical problems and new challenges. *Statistica Sinica*, 11(2):303–351.

Lamoureux, C. G. and Lastrapes, W. D. (1990). Persistence in variance, structural change, and the GARCH model. *Journal of Business & Economic Statistics*, 8(2):225–34.

Lavielle, M. (2005). Using penalized contrasts for the change-point problem. *Signal Processing*, 85(8):1501 – 1510.

Lavielle, M. and Lebarbier, E. (2001). An application of MCMC methods for the multiple change-points problem. *Signal Processing*, 81(1):39 – 53. Special section on Markov Chain Monte Carlo (MCMC) Methods for Signal Processing.

Lavielle, M. and Moulines, E. (2000). Least-squares estimation of an unknown number of shifts in a time series. *Journal of Time Series Analysis*, 21(1):33–59.

Lavielle, M. and Teyssière, G. (2006). Detection of multiple change-points in multivariate time series. *Lithuanian Mathematical Journal*, 46(3):287–306.

Lebarbier, E. (2005). Detecting multiple change-points in the mean of gaussian process by model selection. *Signal Processing*, 85(4):717–736.

Lee, C.-B. (1996). Nonparametric multiple change-point estimators. *Statistics and Probability Letters*, 27(4):295 – 304.

Levin, B. and Kline, J. (1985). The cusum test of homogeneity with an application in spontaneous abortion epidemiology. *Statistics in Medicine*, 4(4):469–488.

Lévy-Leduc, C., Delattre, M., Mary-Huard, T., and Robin, S. (2014). Two-dimensional segmentation for analyzing Hi-C data. *Bioinformatics*, 30(17):386–392.

Lévy-Leduc, C. and Roueff, F. (2009). Detection and localization of change-points in high-dimensional network traffic data. *The Annals of Applied Statistics*, 3(2):637–662.

Li, S. and Lund, R. (2012). Multiple changepoint detection via genetic algorithms. *Journal of Climate*, 25(2):674–686.

Lin, N. and Xi, R. (2011). Aggregated estimating equation estimation. *Statistics and its Interface*, 1(4):73–83.

Liu, J. S. and Lawrence, C. E. (1999). Bayesian inference on biopolymer models. *Bioinformatics*, 15(1):38–52.

Lu, L. and Zhang, H.-J. (2002). Speaker Change Detection and Tracking in Real-Time News Broadcasting Analysis. *Proceedings of the tenth ACM international conference on multimedia*, pages 602–610.

Lung-Yut-Fong, A., Lévy-Leduc, C., and Cappé, O. (2012). Homogeneity and change-point detection tests for multivariate data using rank statistics. *arXiv e-prints 1107.1971*.

Luong, T. M., Rozenholc, Y., and Nuel, G. (2012). Fast estimation of posterior probabilities in change-point models through a constrained hidden Markov model. *arXiv e-prints 1203.4394*.

Maboudou, E. M. and Hawkins, D. M. (2009). Fitting multiple change-point models to a multivariate gaussian model. In *Proceedings of the Second International Workshop in Sequential Methodologies (IWSM 2009)*, pages 1–5.

Maboudou-Tchao, E. M. and Hawkins, D. M. (2013). Detection of multiple change-points in multivariate data. *Journal of Applied Statistics*, 40(9):1979–1995.

Mackey, L., Talwalker, A., and Jordan, M. I. (2013). Divide-and-Conquer Matrix Factorization. *arXiv e-prints 1107.0789v7*.

Maidstone, R., Hocking, T., Rigaill, G., and Fearnhead, P. (2017). On Optimal Multiple Changepoint Algorithms for Large Data. *Statistics and Computing (To Appear)*, 27(2):519–533.

Matloff, N. (2016). Software alchemy: Turning complex statistical computations into embarrassingly-parallel ones. *Journal of Statistical Software*, 71(1):1–15.

Matteson, D. S. and James, N. A. (2014). A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*, 109(505):334–345.

Mood, A. M. (1954). On the asymptotic efficiency of certain nonparametric two-sample tests. *The Annals of Mathematical Statistics*, 25(3):514–522.

Nam, C. F. H., Aston, J. A. D., Eckley, I. A., and Killick, R. (2015). The Uncertainty of Storm Season Changes: Quantifying the Uncertainty of Autocovariance Changepoints. *Technometrics*, 57(2):194–206.

Nam, C. F. H., Aston, J. A. D., and Johansen, A. M. (2012). Quantifying the uncertainty in change points. *Journal of Time Series Analysis*, 33(5):807–823.

Nason, G., von Sachs, R., and Kroisandt, G. (2000). Wavelet processes and adaptive estimation of the evolutionary wavelet spectrum. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 62(2):271–292.

Nikol'skii, I. M. and Furmanov, K. K. (2016). Parallel algorithm to detect structural changes in time series. *Computational Mathematics and Modeling*, 27(2):247–253.

Olshen, A. B., Venkatraman, E. S., Lucito, R., and Wigler, M. (2004). Circular binary segmentation for the analysis of array-based dna copy number data. *Biostatistics*, 5(4):557–572.

Ombao, H. C., Raz, J. A., von Sachs, R., and Malow, B. A. (2001). Automatic statistical analysis of bivariate nonstationary time series. *Journal of the American Statistical Association*, 96(454):543–560.

Page, E. (1954). Continuous inspection schemes. *Biometrika*, 41(1):100–115.

Pein, F., Sieling, H., and Munk, A. (2015). Heterogeneuous change point inference. *arXiv preprints 1505.04898*.

Pepelyshev, A. and Polunchenko, A. S. (2015). Real-time financial surveillance via quickest change-point detection methods. *Statistics and its Inferface*, 0(1):1–14.

Pettitt, A. (1979). A non-parametric approach to the change-point problem. *Applied statistics*, 28(2):126–135.

Picard, F., Hoebacke, M., Lebarbier, E., Miele, V., Rigaill, G., and Robin, S. (2016). *cghseg: Segmentation Methods for Array CGH Analysis*. R package version 1.0.2-1.

Picard, F., Robin, S., Lavielle, M., Vaisse, C., and Daudin, J.-J. (2004). A statistical approach for array CGH data analysis. *BMC Bioinformatics*, 6(27):1–14.

Pickering, B. J. (2015). *Changepoint Detection for Acoustic Sensing Signals*. PhD thesis, Lancaster University.

Reeves, J., Chen, J., Wang, X. L., Lund, R., and Lu, Q. Q. (2007). A review and comparison of changepoint detection techniques for climate data. *Journal of Applied Meteorology and Climatology*, 46(6):900–915.

Rigaill, G. (2015). Pruned dynamic programming for optimal to recover the best segmentations with 1 to k_{max} change-points. *arXiv preprint 1004.0887*.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Roberts, S. W. (2000). Control chart tests based on geometric moving averages. *Technometrics*, 42(1):97–101.

Ross, G. and Adams, N. M. (2012). Two nonparametric control charts for detecting arbitrary distribution changes. *Journal of Quality Technology*, 44(2):102–116.

Ross, G. J., Tasoulis, D. K., and Adams, N. M. (2011). Nonparametric monitoring of data streams for changes in location and scale. *Technometrics*, 53(4):379–389.

Ross, G. J., Tasoulis, D. K., and Adams, N. M. (2013). Sequential monitoring of a bernoulli sequence when the pre-change parameter is unknown. *Computational Statistics*, 28(2):463–479.

Schmidberger, M., Morgan, M., Eddelbuettel, D., Yu, H., Tierney, L., and Mansmann, U. (2009). State of the art in parallel computing with R. *Journal of Statistical Software*, 31(1):1–27.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Scott, A. J. and Knott, M. (1974). A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 30(3):pp. 507–512.

Siegmund, D., Yakir, B., and Zhang, N. R. (2011). Detecting simultaneous variant intervals in aligned sequences. *The Annals of Applied Statistics*, 5(2A):645–668.

Song, Q. and Liang, F. (2015). A split-and-merge Bayesian variable selection approach for ultrahigh dimensional regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(5):947–972.

Srivastava, M. S. and Worsley, K. J. (1986). Likelihood ratio tests for a change in the multivariate normal mean. *Journal of the American Statistical Association*, 81(393):199–204.

Staudacher, M., Telser, S., Amann, A., Hinterhuber, H., and Ritsch-Marte, M. (2005). A new method for change-point detection developed for on-line analysis of the heart beat variability during sleep. *Physica A: Statistical Mechanics and its Applications*, 349(3):582–96.

Stephens, D. A. (1994). Bayesian retrospective multiple-changepoint identification. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 43(1):159–178.

Tartakovsky, A. G., Polunchenko, A. S., and Sokolov, G. (2013). Efficient computer network anomaly detection by changepoint detection methods. *IEEE Journal of Selected Topics in Signal Processing*, 7(1):4–11.

Teyssière, G. (2003). *Interaction Models for Common Long-Range Dependence in Asset Prices Volatility*, pages 251–269. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tibshirani, R. J. and Taylor, J. (2011). The solution path of the generalized lasso. *The Annals of Statistics*, 39(3):1335 – 1371.

Tierney, L., Rossini, A. J., Li, N., and Sevcikova, H. (2015). *snow: Simple Network of Workstations*. R package version 0.4-1.

Tsuchiyama, R., Nakamura, T., Iizuka, T., Asahara, A., Miki, S., Tagawa, S., and Tagawa, S. (2010). *The OpenCL Programming Book*. Fixstars Corporation, 1 edition.

Tsung, F. and Wang, K. (2010). Adaptive charting techniques: Literature review

and extensions. In *Frontiers in Statistical Quality Control 9*, pages 19–35. Physica-Verlag HD, Heidelberg.

Venkatraman, E. S. and Olshen, A. B. (2007). A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics*, 23(6):657–663.

Vert, J.-P. and Bleakley, K. (2010). Fast detection of multiple change-points shared by many signals using group lars. In Lafferty, J., Williams, C., Shawe-taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 2343–2351.

Vostrikova, L. J. (1981). Discovery of "discord" in multidimensional random processes. *Dokl. Akad. Nauk SSSR*, 259(2):270–274.

Wang, H., Killick, R., and Fu, X. (2014). Distributional change of monthly precipitation due to climate change: comprehensive examination of dataset in southeastern united states. *Hydrological Processes*, 28(20):5212–5219.

Wu, H., Salzberg, B., and Zhang, D. (2004). Online event-driven subsequence matching over financial data streams. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 23–34, New York, NY, USA. ACM.

Xi, R., Lin, N., and Chen, Y. (2009). Compression and aggregation for logistic regression analysis in data cubes. *IEEE Transactions on Knowledge and Data Engineering*, 21(4):479–492.

Xie, Y. and Siegmund, D. (2013). Sequential multi-sensor change-point detection. *The Annals of Statistics*, 41(2):670–692.

Yan, G., Xiao, Z., and Eidenbenz, S. (2008). Catching instant messaging worms with change-point detection techniques. In *Proceedings of the 1st Usenix*

Workshop on Large-Scale Exploits and Emergent Threats, LEET'08, pages 6:1–6:10, Berkeley, CA, USA. USENIX Association.

Yao, Y.-C. (1988). Estimating the number of change-points via Schwarz' criterion. *Statistics & Probability Letters*, 6(3):181–189.

Zhang, J. (2002). Powerful goodness-of-fit tests based on the likelihood ratio. *Journal of the Royal Statistical Society Series B*, 64(2):281–294.

Zhang, N. R. and Siegmund, D. O. (2007). A modified Bayes information criterion with applications to the analysis of comparative genomic hybridization data. *Biometrics*, 63(1):22–32.

Zhang, N. R., Siegmund, D. O., Ji, H., and Li, J. Z. (2010). Detecting simultaneous changepoints in multiple sequences. *Biometrika*, 97(3):631–645.

Zhou, H. and Lange, K. (2013). A path algorithm for constrained estimation. *Journal of Computational and Graphical Statistics*, 22(2):261–283.

Zou, C., Liu, Y., Qin, P., and Wang, Z. (2007). Empirical likelihood ratio test for the change-point problem. *Statistics & Probability Letters*, 77(4):374 – 382.

Zou, C., Yin, G., Feng, L., and Wang, Z. (2014). Nonparametric maximum likelihood approach to multiple change-point problems. *The Annals of Statistics*, 42(3):970–1002.

Zou, C. and Zhange, L. (2014). *nmcdr: Non-parametric Multiple Change-Points Detection*. R package version 0.3.0.