Novel Methods for Efficient Changepoint Detection

Gaetano Romano, BSc, MSc
Department of Mathematics and Statistics
Lancaster University

Submitted for the degree of
Doctor of Philosophy

November, 2021

Declaration & Contribution Statements

This thesis has not been submitted, either in whole or in part, for a degree at this, or any other university. I declare that the work presented in this thesis is, to the best of my knowledge and belief, original and my own work. Listed below, a description of the status of each paper forming the main corpus of this thesis and the external contributions to the sections I did not directly contribute to that were left in place for completeness.

The DeCAFS procedure, presented in Chapter 3 is joint work with Paul Fearnhead, Guillem Rigaill and Vincent Runge and has appeared in the Journal of the American Statistical Association (Romano et al., 2021). My contribution to the work was in developing the method, implementing the method within an R package, performing the detailed simulation study, input into methods for estimating the tuning parameters and writing the paper. The theoretical properties introduced in Section 3.5 were obtained by Paul Fearnhead, theory for the infimal convolution was developed by Guillem Rigaill and Vincent Runge, and the results on the application in Section 3.7 were obtained by Guillem Rigaill.

The FOCuS procedure, in Chapter 4 is joint work with Idris Eckley, Paul Fearnhead and Guillem Rigaill. The procedure has been submitted to the Journal of Machine Learning Research. My contribution to the work was in developing the method, implementing the method within an R package, performing both the simulation and the real data studies. Theorem 5, deriving a bound on the expected number of quadratics stored by FOCuS, was obtained by G. Rigaill, as well as its empirical evaluation in Appendix B.3.4.

NUNC, introduced in Chapter 5 has been submitted to Computational Statistics and Data Analysis, where it is currently under review. The manuscript is a collaboration with my fellow PhD student, Edward Austin, and our supervisors Idris Eckley and Paul Fearnhead. Our contributions to the paper are equal with myself leading the more computational aspects of the work, whilst E. Austin focused on the theoretical and application-focused aspects of the paper.

Gaetano Romano

Abstract

This thesis introduces several novel computationally efficient methods for offline and online changepoint detection. The first part of the thesis considers the challenge of detecting abrupt changes in scenarios where there is some autocorrelated noise or where the mean fluctuates locally between the changes. In such situations, existing implementations can lead to substantial overestimation of the number of changes. In response to this challenge, we introduce DeCAFS, an efficient dynamic programming algorithm to deal with such scenarios. DeCAFS models local fluctuations as a random walk process and autocorrelated noise as an AR(1) process. Through theory and empirical studies we demonstrate that this approach has greater power at detecting abrupt changes than existing approaches.

The second part of the thesis considers a practical, computational challenge that can arise with online changepoint detection within the real-time domain. We introduce a new procedure, called FOCuS, a fast online changepoint detection algorithm based on the simple Page-CUSUM sequential likelihood ratio test. FOCuS enables the online changepoint detection problem to be solved sequentially in time, through an efficient dynamic programming recursion. In particular, we establish that FOCuS outperforms current state-of-the-art algorithms both in terms of efficiency and statistical power, and can be readily extended to more general scenarios.

The final part of the thesis extends ideas from the nonparametric changepoint detection literature to the online setting. Specifically, a novel algorithm, NUNC, is introduced to perform an online detection for changes in the distribution of real-time data. We explore the properties of two variants of this algorithm using both simulated and real data examples.

Acknowledgements

This PhD has been to me an incredible journey, full of valuable moments, interesting challenges and remarkable people.

First and foremost, there are no words to fully convey my gratitude towards Idris Eckley and Paul Fearnhead, my PhD supervisors. For providing guidance throughout this journey, for your dedication to review my progress, for supporting me over the asperities that arose over this period, the sincerest thanks to both of you.

The works presented in this thesis would not have been possible without my direct collaborators Edward Austin, Guillem Rigaill and Vincent Runge. I am grateful for the numerous exchanges we had that shaped my research, and I do sincerely hope that there will be the occasion to tackle more challenges together in the future.

Many thanks to Daniel Grose for guiding me through the circles of C++[1].

I would like to thank and acknowledge all the numerous friends and colleagues from the StatScale office, the Department and Lancaster University for the valuable discussions we held together[2].

I wish to thank Pierre Nicholas for providing data for the application covered in Chapter 3.

I would like to thank Franck Picard, the external examiner, and Rebecca Killick, the internal examiner, for reviewing this work, and for the stimulating discussion that we held in the viva. This provided valuable comments and ideas that will certainly benefit to this and my future works.

Many thanks to Lancaster University and the EPSRC (EP/N031938/1 StatScale grant) for the support that made the work presented in this thesis possible.

On a more personal basis, I would like to thank the numerous people close to me over these three years[3]. Thanks for the precious moments we had over this period, either physically or remotely. In a way or another, you all have all been of great help.

Lastly, to My sister Chiara, My mother Maria Pia and My father Giuseppe to whom I owe everything, thank you.

[1] Like Virgil: "Fede nei puntator esser sì bassa\\ l'error di segmentazion ivi regna\\ non ragionam di lor ma guarda e passa".

[2] As many of you may know already, my coffee machine takes a moment to heat up: consider this an invitation.

[3] And apologies for not being able to thank you directly, but (luckily for me) you are far too many to be mentioned here. As denoted by Eco (1995), writing some comprehensive Acknowledgments is a rather challenging task, far too complex to be resolved optimally the evening before the submission.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Changepoint detection – also commonly known as change detection or sometimes as break-point detection – is the statistical analysis that focuses on where (or whether) one or more changes in some measurable properties of a temporal or spatial process occurred. For simplicity, let us consider an introductory example, taking a departure from this vague definition. In Figure 1.1 we find a realization of a piecewise stationary Gaussian process: the observations are centered on a piecewise constant signal with fixed variance. Two changes are present respectively at time 1000 and 3000, segmenting the sequence into three subsets. Inferring such locations, as well as the within-segment parameters, is the goal of our analysis.

Albeit being of historical interest (with seminal works such as Page, 1954; Scott and Knott, 1974; Eiauer and Hackl, 1978), the problem of detecting change points has seen increasing popularity over the last decade. Recent technological developments, an ever-growing amount of large data streams – often intractable by traditional techniques – have increased the demand for fast and efficient change point detection algorithms (among many, we mention Killick et al., 2012; Maidstone et al., 2017; Fryzlewicz, 2014; Eichinger and Kirch, 2018). Lowering the computational complexity as much as possible has proved to be extremely important for many applications from the industry and several fields such as medicine, neuroscience, genomics, astrophysics, *etc.*

The work included in this thesis was motivated by the need of extending those fast procedures to more non-standard scenarios – often present in practical applications – which could pose a challenge to existing methodologies either of statistical or of computational nature. Three novel fast and efficient changepoint detection procedures are therefore presented.

Chapter 3 introduces DeCAFS, a novel recursion of a model-based approach that extends the FPOP procedure (Rigaill, 2015) to data where the usual i.i.d. assumptions fail, accounting for both autocorrelation in the noise and local fluctuations in the

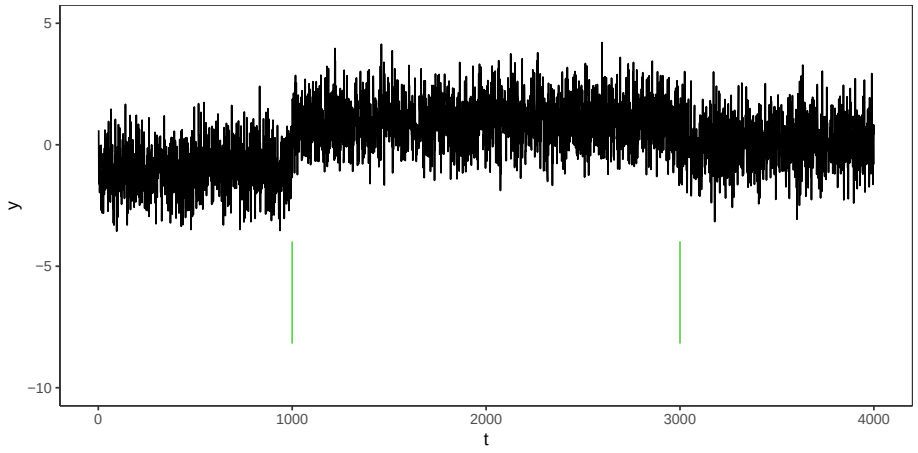


Figure 1.1: A realization of a Gaussian process with 2 changes in the mean parameter, flagged by two vertical segments in green.

signal object of interest, while still retaining the desirable log-linear computational complexity. DeCAFS, is capable of outperforming the current state-of-the-art algorithms in a variety of real-world scenarios.

Chapters 4 and 5 focus on real-time analysis of a data stream, that we call online changepoint detection. From the introductory example, the difference is in that we have not observed the entire sequence, and we wish to test as we obtain new observations whether a change has occurred or not. In this domain, several challenges appear that render most of the current methods infeasible. In addition to the unbounded nature of the data, which does not allow for any pre-processing nor infinite computations and memory, a glimpse at the collection of real-world problems featured in Ahmad et al. (2017) would clarify the need for novel procedures. In particular FOCuS, from Chapter 4, takes advantage again of a recursion similar to the FPOP recursion, solving exactly the established sequential Page-CUSUM statistics for Gaussian change-in-mean. In this way, FOCuS improves on related methodologies either in terms of statistical power or computational efficiency. Within the same Chapter is presented an application on monitoring AWS Cloudwatch CPU utilization. Lastly, introduced in Chapter 5, NUNC, is an algorithm for online nonparametric changepoint detection. This work was motivated by an open challenge posed by an industrial partner, that of detecting anomalous behaviour in telecommunication data, however proved itself to be efficient even on a different application.

Preceding the three methodological chapters, Chapter 2 provides a brief review of the current state-of-the-art. This provides an introduction for the reader of some basic knowledge of changepoint detection necessary to access the rest of the thesis, should they not be acquainted with the field.

Chapter 2

Literature Review

Numerous approaches to changepoint detection have been introduced over recent years. Despite all tackling a common problem, they differ in terms of approach or formulation. Performing changepoint detection on a given sequence of observations could correspond to either directly estimating the underlying generating signal and the exact change locations; finding the optimal segmentation of the data – effectively clustering an ordered set of elements; or testing whether a change or an anomalous behaviour has been observed over a subset of the observations. Several classifications of changepoint methods can be made. Some of these focus on the nature of the analysis, separating univariate from multivariate, *offline* from sequential or *online* procedures; other classifications discriminate the literature based on the underlying model assumptions or on the nature of the change to estimate. What follows is a classification that organises procedures based on the algorithmic approach employed to solve the changepoint problem: such a classification will hopefully provide some insights on how the various procedures relate in terms of the various optimization problems they solve, from a computational perspective. In Section 2.1 we introduce approaches that are based upon dynamic programming (in particular the constrained and penalised approach), in Section 2.2 we focus on those methods based upon the divide-and-conquer Binary Segmentation approach, in Section 2.3 we briefly review a selection of other approaches within the literature.

2.1 Constrained and Penalised Approaches

Constrained and penalised algorithms form a family of multiple-changepoint detection algorithms. They operate via specifying a cost function for a given segmentation and recursively seeking for the segmentation that achieves the smallest cost. The overview section mostly takes inspiration from the work of Maidstone et al. (2017).

2.1.1 An overview of the constrained and penalised approaches

To provide a common framework, let $y_1, \dots y_n$ be an ordered sequence of observations. Across the course of this thesis, to denote the subset of such a sequence from s to t, we will write $y_{s:t} = y_s, \dots, y_t$ for $s < t$. By assuming that the sequence can be split into $K+1$ segments, we denote with $\tau_0, \tau_1, \dots, \tau_K, \tau_{K+1}$ the set of ordered changepoints, where $\tau_0 = 0$, $\tau_{K+1} = n$ by definition and $\tau_k < \tau_{k+1}\ \forall\ k \in 1, ..., K$. A segment is a subset of the data ranging between between two contiguous changepoints, in our notation, the segment between $\tau_k + 1$ and τ_{k+1} will therefore be denoted by $y_{\tau_k+1:\tau_{k+1}}$.

The objective is to find the optimal segmentation of our sequence $y_{1:n}$. This consists of finding the best set of $\tau = \tau_0, \dots, \tau_{K+1}$ changepoints that minimise some cost:

$$\min_{\substack{K\in\mathbb{N} \\ \tau_1,\dots,\tau_K}} \sum_{k=0}^{K} \mathcal{L}(y_{\tau_k+1:\tau_{k+1}}), \tag{2.1}$$

where $\mathcal{L}(y_{s+1:t})$ is the cost for a segment for points $y_{s+1}, \dots y_t$. If we assume the data is independent and identically distributed within each segment, for segment parameter θ, then this cost can be obtained through:

$$\mathcal{L}(y_{s+1:t}) = \min_{\theta} \sum_{i=s+1}^{t} -\log(f(y_i, \theta)) \tag{2.2}$$

with $f(y, \theta)$ being the likelihood for data point y if the segment parameter is θ. Minimizing this corresponds to finding multiple changes in the underlying parameter θ. The form of $\mathcal{L}(\cdot)$ depends both on the likelihood of choice and on the piecewise constant parameters objects of inference. Some of the most common loss functions can be found in Table 2.1, however these can vary accordingly to the nature of the change to estimate, as we are going to see in the next subsections.

The issue with such optimization problems lies in the fact that the optimal segmentation would always be the one that separates each observation into a single cluster, *i.e.* $\hat{K} = n$, $\hat{\tau} = 0, 1, \dots, n$. A constraint needs therefore to be added to get a sensible estimate of the number of changepoints. Two major approaches are possible: solving a penalised optimization problem, originating from Yao (1988), and solving a constrained optimization problem, which was first introduced with Yao and Au (1989).

The constrained optimization problem. One possible solution is to solve 2.1 for a fixed value of K. The segment neighbourhood procedure estimates the optimal cost for segmenting a sequence of n observations in $K+1$ parts:

$$\mathcal{Q}_n^{(K)} = \min_{\tau_1,\dots,\tau_K} \sum_{k=0}^{K} \mathcal{L}(y_{\tau_k+1:\tau_{k+1}}).$$

Model	Parameter	$\mathcal{L}(y_{s+1:t})$
$y_{s+1:t} \sim \mathcal{N}(\mu, \sigma^2)$	μ	$\frac{1}{2\sigma^2}\sum_{i=s+1}^{t}(y_i - \bar{y}_{s+1:t})^2$
$y_{s+1:t} \sim \mathcal{N}(\mu, \sigma^2)$	σ	$(t-s)\left[\log(\frac{\sum_{i=s+1}^{t}(y_i-\mu)^2}{t-s}) + 1\right]$
$y_{s+1:t} \sim \mathcal{N}(\mu, \sigma^2)$	$\mu,\ \sigma$	$(t-s)\left[\log(\frac{\sum_{i=s+1}^{t}(y_i-\bar{y}_{s+1:t})^2}{t-s}) + 1\right]$
$y_{s+1:t} \sim \text{Pois}(\lambda)$	λ	$(1 + \log(\bar{y}_{s+1:t}))\sum_{i=s+1}^{t} y_i$

Table 2.1: Some loss functions for different changepoint optimization problems. The first column gives the underlying model assumption, the second column the parameter object of inference, the third the correspondent loss. For brevity reasons, we write $\bar{y}_{l:u}(u-l+1) = \sum_{j=l}^{u} y_j$.

Segment Neighbourhood takes advantage of a dynamical programming recursion to solve this non-convex optimization problem with computational complexity $K\mathcal{O}(n^2)$. This is achieved through the recursion:

$$\mathcal{Q}_t^{(K)} = \min_{\tau<t}\left[\mathcal{Q}_\tau^{(K-1)} + \mathcal{L}(y_{\tau+1:t})\right].$$

For a fixed $K < n$ it is possible to compute this recursion for all $t = 1, \ldots, n$. The quadratic increase in computational complexity comes from the fact that at each iteration a check is needed for all $\tau = 1, \ldots, t-1$ to perform the minimization.

This approach can be beneficial when the exact number of changes in the sequence is known *a priori*. However, when K is unknown Maidstone et al. (2017) suggest fixing a maximum number of changes allowed K, computing $\mathcal{Q}_n^{(k)}$ for all $k = 0, 1, \ldots, K$, and then minimizing for $\mathcal{Q}_n^{(k)} + g(k, n)$ for some penalty function $g(k, n)$:

$$\min_k\left[\mathcal{Q}_n^{(k)} + g(k, n)\right]$$

The penalised optimization problem. When the penalty function is linear in k, *i.e.* $g(k, n) = \beta k$, for $\beta > 0$ (and β potentially depending on n) we can write:

$$\mathcal{Q}_{n,\beta} = \min_k\left[\mathcal{Q}_n^{(k)} + \beta k\right] = \min_{k,\tau}\left[\sum_{k=0}^{K}\mathcal{L}(y_{\tau_k+1:\tau_{k+1}}) + \beta\right] - \beta. \tag{2.3}$$

This is known as the penalised cost optimization problem. The cost function $\mathcal{Q}_{n,\beta}$ represent the optimal cost for segmenting the data up to time n given a penalty β.

Remark 1 *The penalised changepoint problem can be formulated as an $\ell 0$ penalised non-convex optimization problem. For simplicity, let us focus on the Gaussian change in mean problem. We observe $y_t \sim N(\mu_t, 1)$, where μ_t is the piecewise constant signal*

object of inference. That is, we find a change at t whether $\mu_t \neq \mu_{t-1}$. *We estimate the number and location of the changepoints by effectively solving:*

$$\min_{\mu_{1:n}} \sum_{t=1}^{n} \left[(y_t - \mu_t)^2 + \beta \mathbb{1}_{\delta_t \neq 0} \right] \tag{2.4}$$

where $\delta_t = \mu_t - \mu_{t-1}$, $\mathbb{1} \in \{0, 1\}$ *is an indicator function,* β *is the penalty for adding a changepoint.*

The Optimal Partitioning (OP) (Jackson et al., 2005) solves 2.3 exactly through a dynamic programming recursion. With $\mathcal{Q}_{0,\beta} = -\beta$, we write the recursion:

$$\mathcal{Q}_{t,\beta} = \min_{0 \leq \tau < t} \left[\mathcal{Q}_{\tau,\beta} + \mathcal{L}(y_{\tau+1:t}) + \beta \right] \tag{2.5}$$

for $t = 1, \ldots, n$. Considering that at each iteration we need to evaluate values for t segmentation costs, we find that OP shows $\mathcal{O}(n^2)$ computational complexity. In Algorithm 1 we summarise the Optimal Partitioning procedure.

For the rest of this thesis we are going to be mostly focusing on the penalised approach. Both approaches are closely related, and extensions that are found for one are easily ported to the other optimization problem. However, the penalised optimization problem incorporates model selection, and possibly for this reason it is the predominant optimization problem in the dynamical programming changepoint literature. To simplify notation we will refer to the optimal penalised cost $\mathcal{Q}_{n,\beta}$ simply as $\mathcal{Q}_n$.

Algorithm 1: Optimal Partitioning

Data: $y_{1:n} = \{y_1, \ldots, y_n\}$ a series of length n
Input: $\beta > 0$
begin Initialisation
  $\mathcal{Q}_0 \leftarrow -\beta$
  $cp(0) \leftarrow \{0\}$
end
for $t = 1$ *to* n **do**
  $\mathcal{Q}_t \leftarrow \min_{0 \leq \tau < t} \left[\mathcal{Q}_\tau + \mathcal{L}(y_{\tau+1:t}) + \beta \right]$
  $\hat{\tau} \leftarrow \operatorname{argmin}_{0 \leq \tau < t} \left[\mathcal{Q}_\tau + \mathcal{L}(y_{\tau+1:t}) + \beta \right]$
  $cp(t) \leftarrow (cp(\hat{\tau}), \hat{\tau})$
end
Return $cp(n)$

Reducing the computational complexity of OP and SN. We mentioned how the OP and SN solutions to the penalised and constrained minimisation have both

quadratic computational complexity in the number of observations. Naïvely, reducing the numbers of checks to be performed at each iteration reduces the complexity, and in some situations it is possible to do so without resorting to an approximation. Performing pruning, while still solving exactly the two optimization problems, is possible if one of the two conditions on the segment costs holds:

Condition 1 *There exists a constant κ such that for every $l < t < u$:*

$$\mathcal{L}(y_{l+1:t}) + \mathcal{L}(y_{t+1:u}) + \kappa \leq \mathcal{L}(y_{l+1:u})$$

Condition 2 *There exist some function $\gamma(\cdot, \theta)$ such that the cost function satisfies:*

$$\mathcal{L}(y_{s+1:t}) = \min_{\theta} \sum_{i=s+1}^{t} \gamma(y_i, \theta)$$

for every $s,\ t$.

Condition 1 is at the basis of the *inequality based pruning* methods, originating from Killick et al. (2012) where the PELT procedure is introduced; Condition 2 is at the basis of the *functional pruning* methods, originating from the pDPA from Rigaill (2010, 2015). PELT performs pruning over the OP recursion, whilst pDPA is based on the SN recursion. Maidstone et al. (2017) showed that it is possible to use inequality based pruning for the constrained optimization problems (the SNIP procedure), and functional pruning within optimal partitioning (the FPOP procedure). Table 2.2 summarises the relationships between these procedures. We now cover in detail the PELT and FPOP procedures.

	No Pruning	Ineq. Based Pruning	Functional Pruning
Penalised Approach	OP	PELT	FPOP
Constrained Approach	SN	SNIP	pDPA

Table 2.2: A map of the base constrained and penalised approaches and their faster implementations.

PELT: inequality based pruning. The PELT algorithm – acronym for Pruned Exact Linear Time – solves exactly the penalised minimization of 2.3 with an expected computational cost that can be linear in n – while still retaining $\mathcal{O}(n^2)$ computational complexity in the worst case. This is achieved by reducing the number of segment costs to evaluate at each iteration via an additional pruning step based on Condition 1. That is, if

$$\mathcal{Q}_\tau + \mathcal{L}(y_{\tau+1:t}) + \kappa \geq \mathcal{Q}_t$$

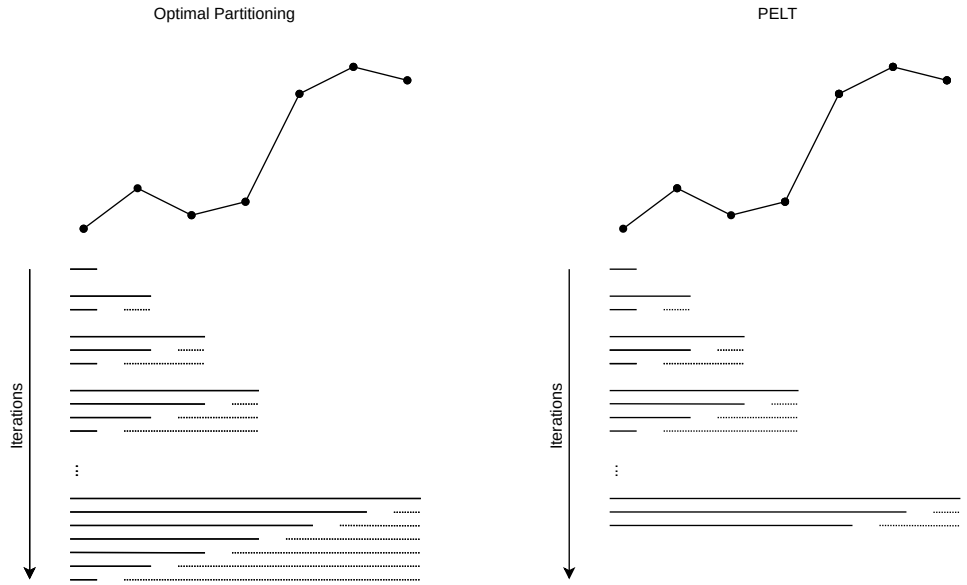


Figure 2.1: An illustration of the OP and PELT recursions. Solid lines correspond to the optimal segmentation costs $\mathcal{Q}_\tau$, whilst dotted lines to the functions $\mathcal{L}(y_{\tau+1:t})$. We can see how the costs to evaluate at each iteration increase linearly in OP, whilst they are reduced in PELT whether the pruning condition is met.

then we can safely prune the segment cost related to τ, as τ will never be the optimal changepoint location up to any time $T > t$ in the future. An illustration of this is given in Figure 2.1, and a summary of the PELT algorithm can be found in Algorithm 2 (assumes $\kappa = \beta$).

There are virtually no disadvantages in implementing the PELT recursion over the OP recursion, as the computational cost is at worst that of OP plus the cost of checking the pruning condition at each iteration. In many situations however the gain in speed over OP can be substantial, and Killick et al. (2012) show that if the number of changes increases linearly with n then PELT can have a computational cost that is linear in n. More generally the computational benefits of PELT are largest when we have many changepoints and short segments.

FPOP: functional pruning. FPOP – acronym for Functional Pruning Optimal Partitioning – solves again the OP minimization in $\mathcal{O}(n \log n)$, worst case $\mathcal{O}(n^2)$ computational complexity. The FPOP recursion breaks down the cost into its functional form, and can be shown to prune more efficiently than PELT – that is at any iteration FPOP will have the same or fewer candidates for the most recent changepoint than PELT. A dedicated comparison of the two pruning methods can be

Algorithm 2: PELT

Data: $y_{1:n} = \{y_1, \ldots, y_n\}$ a series of length n
Input: $\beta > 0$
begin Initialisation
  $\mathcal{Q}_0 \leftarrow -\beta$
  $cp(0) \leftarrow \{0\}$
  $R_1 = \{0\}$
end
for $t = 1, \ldots, n$ **do**
  $\mathcal{Q}_t \leftarrow \min_{\tau \in R_t} [\mathcal{Q}_\tau + \mathcal{L}(y_{\tau+1:t}) + \beta]$
  $\hat{\tau} \leftarrow \text{argmin}_{\tau \in R_t} [\mathcal{Q}_\tau + \mathcal{L}(y_{\tau+1:t}) + \beta]$
  $cp(t) \leftarrow (cp(\hat{\tau}), \hat{\tau})$
  $R_{t+1} \leftarrow \{\tau \in \{R_t \cup \{t\}\} : \mathcal{Q}_\tau + \mathcal{L}(y_{\tau+1:t}) + \beta \leq \mathcal{Q}_t\}$
end
Return $cp(n)$

found in Section 6 of Maidstone et al. (2017).

In terms of the methodology, the idea behind functional pruning consists of splitting the cost function $\mathcal{L}(\cdot)$ into its core components $\gamma(\cdot, \theta)$ according to Condition 2. Starting from the OP cost recursion we obtain:

$$\begin{aligned}
\mathcal{Q}_t &= \min_{0 \leq \tau < t} [\mathcal{Q}_\tau + \mathcal{L}(y_{\tau+1:t}) + \beta] \\
&= \min_{0 \leq \tau < t} \left[\mathcal{Q}_\tau + \min_\theta \sum_{i=\tau+1}^{t} \gamma(y_i, \theta) + \beta \right] \\
&= \min_\theta \min_{0 \leq \tau < t} \left[\mathcal{Q}_\tau + \sum_{i=\tau+1}^{t} \gamma(y_i, \theta) + \beta \right] \\
&= \min_\theta \min_{0 \leq \tau < t} q_t^\tau(\theta) \\
&= \min_\theta Q_t(\theta),
\end{aligned}$$

with $q_t^\tau(\theta)$ being the optimal cost of partitioning the data up to time t conditional on the last changepoint being at τ and the current segment parameter being θ, while $Q_t(\theta)$ is the optimal cost of partitioning the data up to time t with the current segment parameter being θ. Then, for $Q_0(\theta) = 0$, it is possible to derive the recursion:

$$Q_t(\theta) = \min \left\{ Q_{t-1}(\theta),\ \min_\theta Q_{t-1}(\theta) + \beta \right\} + \gamma(y_t, \theta). \tag{2.6}$$

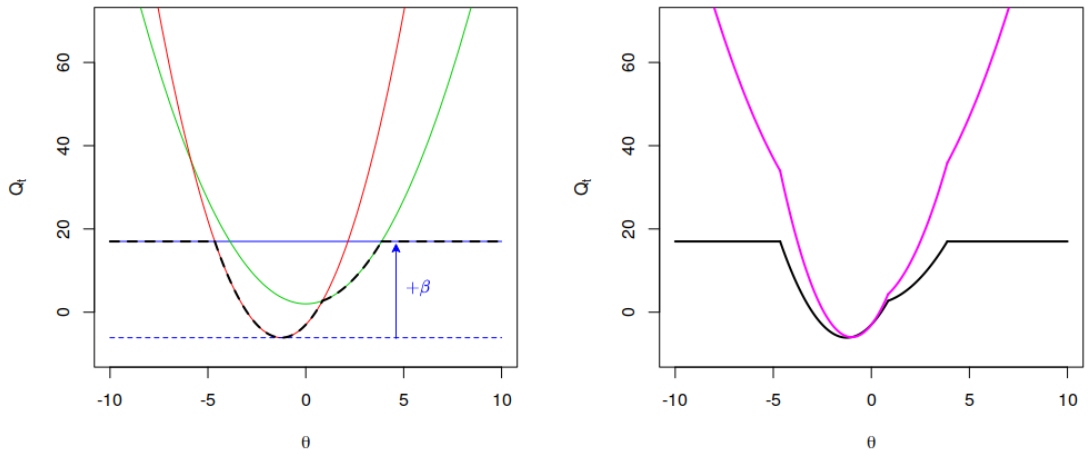
Here, $Q_t(\theta) = \min_\tau q_t^\tau(\theta)$ presents itself in the form of a piecewise function in θ, where $q_t^\tau(\theta)$ are its components. Performing the minimization over τ consists in reconstructing the domain where each component $q_t^\tau(\theta)$ is optimal, *i.e.* finding $\mathcal{D}_\tau \subseteq \mathbb{R} : \theta \in \mathcal{D}_\tau \iff q_t^\tau(\theta) = Q_t(\theta)$. If a component function is never optimal for any θ, we can simply prune it. A description of the FPOP procedure can be found in Algorithm 3. For a graphical representation of the recursion see Figure 2.2.

Algorithm 3: FPOP

Data: $y_{1:n} = \{y_1, \dots, y_n\}$ a series of length n
Input: $\beta > 0$.
begin Initialisation
  $Q_0(\theta) \leftarrow 0$
  $\hat{\theta}_0 \leftarrow 0$
end
for $t = 1, \dots, n$ **do**
  $Q_t(\theta) \leftarrow \min\{Q_{t-1}(\theta),\ \min_\theta Q_{t-1}(\theta) + \beta\} + \gamma(y_t, \theta)$
  $\hat{\theta}_t \leftarrow \operatorname{argmin} Q_t(\theta)$
  if $\gamma(\hat{\theta}_t, \hat{\theta}_{t-1}) > \beta$ **then**
    $\hat{\tau} \leftarrow (t, \hat{\tau})$
  end
end
Return $\hat{\theta}_{1:n}$, $\hat{\tau}$

The major drawback of FPOP compared to PELT, and of all functional pruning methods in general, is that in several cases it is only possible to solve the recursion for a one-dimensional θ parameter. Furthermore, writing an implementation for the above-mentioned recursion is often non-trivial as the form of $Q_t(\theta)$ depends on the underlying model assumptions. For example, in the simple Gaussian change-in-mean setting, with θ being the mean parameter object of inference, we find that $\gamma(y_t, \theta) = (y_t - \theta)^2$ is quadratic in θ, therefore each $q_t^\tau(\theta)$ will be a quadratic and $Q_t(\theta)$ will be a piecewise quadratic in θ. Other update functions may be more difficult to handle, such as the Poisson change-in-paramter, where the cost function is $\gamma(y_t, \theta) = (y_t \log\theta) - \theta$ has both terms in θ and $\log\theta$: in such case the intersections of the piecewise functions cannot be expressed in terms of elementary functions. On the positive side, working directly with the functional form of the cost can allow for adaptations of the penalised optimization problem to more sophisticated models and scenarios, as we are going to see in the next paragraphs.

On a side note, two forms of the FPOP recursion are present in the literature. The one in (2.6) originates from Hocking et al. (2017): while being slightly different from



(a) (b)

Figure 2.2: A representation of a single iteration of the FPOP procedure. In (a), 3 quadratics $q_t(\theta)$, with the two in red and the green being the quadratics for a change respectively at $\tau = 1, 2$; in blue, the line obtained from $\min_\theta Q_{t-1}(\theta) + \beta$; the dotted black line is the optimal cost obtained from the pruning step. In (b), the update step $(y_t - \theta)^2$ for adding the latest point y_t, from the pruned cost, in black, to the final updated cost $Q_t(\theta)$, in purple.

the one reported in Maidstone et al. (2017), it shares the same concepts but is more compact as it maps a piecewise constant function to a piecewise constant function, incorporating both the minimization, the pruning and the update of the optimal cost. For this reason, it is the one adopted by many recent works related to functional pruning (e.g. Runge et al., 2020a; Jewell et al., 2020), as well as the one employed in both Chapter 2 and Chapter 3 of this thesis.

Choosing the $\ell 0$ penalty. Concerning the penalised optimization problem, some research focused on how to pick sensible values for the penalty β. One such of these works, Lavielle and Moulines (2000), provides theoretical guarantees for consistency of various common penalties. As an alternative to a fixed value, in Haynes et al. (2017a), CROPS, is introduced: a sequential procedure to find the optimal segmentation of a sequence of observations over a range of penalties.

2.1.2 Extensions to different models

We now proceed by listing some of the numerous model extensions that can be applied to the penalised and constrained recursions.

Nonparametric PELT. To perform nonparametric changepoint detection, the NP-PELT procedure, from Haynes et al. (2017b) uses a segment cost chosen to detect substantial changes in the empirical cumulative distribution function. This procedure is detailed in Chapter 5, as the procedure shared in that chapter employs the same nonparametric cost.

Change in slope of a linear model. Two works focused on detecting changes in slope of a linear process through the penalised optimization problem: Fearnhead et al. (2018) and Runge et al. (2020b). They both specify a similar cost function:

$$\mathcal{L}(y_{(\tau+1):t}, s, e) = \sum_{i=\tau+1}^{t} \left(y_i - s + (i-\tau)\frac{e-s}{t-\tau} \right)^2,$$

which performs linear interpolation of a linear function that takes the value s at time τ and e at time t. The minimization is performed, in CPOP, through a penalised minimization problem with constraints on the segment length, solved through a functional pruning recursion, and in SlopeOP over a finite set of real values $s, e \in \mathcal{S}$ of size m. Both algorithms achieve similar performances: a detailed comparison of the two can be found in the more recent work Runge et al. (2020b).

Change in the AR autocovariance, AR noise. In Section 4.3 of Killick et al. (2012) a cost function for detecting changes in the autocovariance is introduced. For detecting abrupt changes in the piecewise stationary signal of a sequence with autocorrelation in the noise, we find the AR1Seg procedure (Chakar et al., 2017): this procedure is based on the functional pruning pDPA recursion. More details about AR1Seg can be found within Section 3.1: Chapter 3 introduces a procedure that solves the generalization of such problem through a penalised functional pruning recursion.

Robust FPOP. Fearnhead and Rigaill (2019) introduce a changepoint procedure that makes the FPOP procedure robust to point-outliers. The idea is to replace the cost $(y_t - \mu)^2$ with a different loss that increases at a slower rate in $|y_t - \mu|$. Different loss functions are presented in the study, of those, we present the Huber loss (from Huber, 2004)

$$\gamma(y_t, \theta) = \min\{(y_t - \theta)^2,\ 2K|y - \theta| - K^2\},$$

and the biweight loss

$$\gamma(y_t, \theta) = \min\{(y_t - \theta)^2,\ K^2\},$$

with the latter being employed in the real-data application of Chapter 4. These are to be implemented in the recursion at the update step.

2.1.3 Extensions to different Change Scenarios

This subsection briefly covers some procedures that extend the simple recursions to more sophisticated change scenarios.

Collective and Point Anomaly Detection. Another changepoint scenario involves situations where there is a normal behaviour, and then segments of time where the features of the data change from this. These segments are known as collective anomalies, and the changes are commonly referred to as epidemic changes. The CAPA procedure of Fisch et al. (2018) extends the PELT recursion to estimate collective anomalies, potentially in the presence of point anomalies. This is achieved through minimizing the penalised cost:

$$\sum_{t \notin \cup [s^i+1, e^i]} -\log f(y_t, \theta^0) + \sum_{j=1}^{K} \left[\min_{\theta^j} \left(\sum_{t=s^j+1}^{e^j} -\log f(y_t, \theta^j) \right) + \beta \right],$$

where $f(\cdot, \cdot)$ is our likelihood, θ^0 is the parameter under the normal behaviour, $(s^1, e^1), \ldots, (s^K, e^K)$ are the collective anomaly intervals, with an anomaly starting at s^j and ending at e^j with parameters $\theta^j \neq \theta^0$, $j = 1, \ldots, K$. From its origin CAPA has been extended to deal with multivariate sequences in Fisch et al. (2019), and to perform online changepoint detection in Fisch et al. (2020).

Constrained FPOP and GFPOP. Hocking et al. (2017) propose an extension of FPOP capable of providing inference on specific change patterns (for instance only detecting up changes). This is achieved through specifying a constraint on the parameters within the functional pruning minimization, deriving the recursion:

$$Q_t(\theta) = \min \left\{ Q_{t-1}(\theta),\ \mathcal{K}_{\theta, Q_{t-1}} + \beta \right\} + \gamma(y_t, \theta),$$

where $\mathcal{K}_{\theta,Q}$ is an operator necessary to constrain a specific type of change. For example, to only detect up changes, our operator will be $\mathcal{K}^{\leq}_{\theta,Q} = \min_{\theta' < \theta} Q(\theta')$. Continuing our example, in the Gaussian change-in-mean, to constrain up changes – often referred to as isotonic regression, the constrained FPOP recursions becomes:

$$Q_t(\mu) = \min \left\{ Q_{t-1}(\mu),\ \min_{\mu' < \mu} Q_{t-1}(\mu') + \beta \right\} + (y_t - \mu)^2.$$

Based on this idea, the GFPOP procedure, from Runge et al. (2020a), develops a general framework for constrained changepoint detection, translating the changepoint problem within a discrete-state hidden Markov model that allows for nonparametric modelling on the shape of the parameters between state transitions.

2.1.4 Other extensions

Parallel PELT. In Tickle et al. (2020) we find two ways to increase the efficiency of the PELT procedure by taking advantage of parallelization. Two algorithms are presented: the first procedure, Chunk, separates the sequence into multiple segments which are allocated to different cores; in the second procedure, Deal, each core evaluates only a subset of the sequence derived from equally spaced points.

2.2 Binary Segmentation approaches

Binary Segmentation (BS), from Scott and Knott (1974) and Sen and Srivastava (1975), is a procedure that aims to segment the sequence through an iterative divide-and-conquer approach. For this reason, BS is often employed to extend single changepoint procedures to multiple changes procedures, and hence it is one of the most prominent methods in the literature.

2.2.1 An overview of the binary segmentation approaches

The idea behind Binary Segmentation is to test for a change by splitting a sequence into two segments and to check if the cost over those two segments is smaller then the cost computed on the whole sequence. For example if our test is based on a log-likelihood ratio statistic, or similar, then we test if there is a τ that satisfy:

$$\mathcal{L}(y_{1:\tau}) + \mathcal{L}(y_{\tau+1:n}) + \beta < \mathcal{L}(y_{1:n}) \tag{2.7}$$

with $\beta \in \mathbb{R}$, and segment cost $\mathcal{L}(\cdot)$, as in 2.2. If the condition in 2.7 is true for at least one $\tau \in 1, \ldots, n$, then the τ that minimizes $\mathcal{L}(y_{1:\tau}) + \mathcal{L}(y_{\tau+1:n})$ is picked as a changepoint and the test is then performed on the two newly generated splits. The procedure is repeated until no further changepoints are detected on all resulting segments. In Algorithm 4 we describe the Binary Segmentation as a recursive procedure, where the first iteration would be simply given by $\text{BinSeg}(y_{1:n}, \beta)$.

In Killick et al. (2012) it is noted that while BS attempts to minimize the same cost as OP in 2.3, it is not guaranteed that it will solve such minimization optimally; as reported in Fryzlewicz (2014), BS is consistent "whether minimum spacing between any two adjacent change-points is of order greater than $n^{3/4}$": for smaller segments lengths BS might miss a change. The major advantage with respect to OP comes from the fact that the BS procedure is of $\mathcal{O}(n \log n)$ computational complexity: this is because BS is a greedy procedure, in the sense that as soon as condition 2.7 is not met, the corresponding segmentation is immediately discarded and never evaluated again. The BS procedure essentially builds a directed tree, where branches point to a search for a split over different subsets of the data. Branches are pruned as soon as

Algorithm 4: Binary Segmentation – BinSeg$(y_{s:t}, \beta)$

Data: $y_{s:t} = \{y_s, \dots, y_t\}$ a series of length $t - s + 1$
Input: $\beta > 0$.

if $t - s \leq 1$ **then**
  Return {};
end
$\mathcal{Q} \leftarrow \min_{\tau \in \{s,\dots,t\}} [\mathcal{L}(y_{s:\tau}) + \mathcal{L}(y_{\tau+1:t}) - \mathcal{L}(y_{s:t}) + \beta]$;
if $\mathcal{Q} < 0$ **then**
  $\hat{\tau} \leftarrow \operatorname{argmin}_{\tau \in \{s,\dots,t\}} [\mathcal{L}(y_{s:\tau}) + \mathcal{L}(y_{\tau+1:t}) - \mathcal{L}(y_{s:t})]$;
  $cp \leftarrow \{\hat{\tau},\ \text{BinSeg}(y_{s:\hat{\tau}}, \beta),\ \text{BinSeg}(y_{\hat{\tau}+1:t}, \beta)\}$;
  Return cp;
end
Return {};

no candidate splits meet the condition. In order to gain statistical power, two major procedures have been introduced in the literature that can improve on the returned segmentation: we illustrate those as follows.

Wild Binary Segmentation. WBS, Fryzlewicz (2014), and its extension WBS2 Fryzlewicz (2020) are multiple changepoints procedures that improve on the BS changepoint estimation via computing the initial segmentation cost of BS multiple times over $M + 1$ random subsets of the sequence, $y_{s_1:t_1}, \dots, y_{s_M:t_M}, y_{1:n}$, picking the best subset according to what achieves the smallest segmentation cost and reiterating the procedure over that sample accordingly. The idea behind WBS lies in the fact that a favourable subset of the data $y_{s_m:t_m}$ could be drawn which contains a true change sufficiently separated from both sides s_m, t_m of the sequence. By the inclusion of the $y_{1:n}$ entire sequence amongst the subsets, it is guaranteed that WBS will do no worse than the simple BS algorithm. An iterative procedure is detailed in Algorithm 5, again, as before, the algorithm is initiated with WildBinSeg$(y_{1:n}, \beta, M)$.

One of the major drawbacks of WBS is that in scenarios where we find frequent changepoints, in order to retain a close-to-optimal estimation, one should draw a higher number of M intervals: this can be problematic given that WBS has computational complexity that grows linearly in the total length of the observations of the subsets.

Seeded Binary Segmentation (SBS). Kovács et al. (2020) introduces the SBS algorithm to mitigate the issues of WBS that arise from searching over random segments of data. The idea is to eliminate the aleatory component from the WBS procedure and to introduce a deterministic interval generating scheme such

Algorithm 5: Wild Binary Segmentation – WildBinSeg($y_{s:t}, \beta, M$)

Data: $y_{s:t} = \{y_s, \dots, y_t\}$ a series of length $t - s + 1$
Input: $\beta > 0$.
1 **if** $t - s \leq 1$ **then**
2 | Return {};
3 **end**
4 Draw $\mathcal{M} = \{[s_1, t_1], \dots, [s_M, t_M]\}$ tuples of subset indexes;
5 $\mathcal{M} \leftarrow \mathcal{M} \cup \{[1, n]\}$
6 $\mathcal{Q} \leftarrow \min_{\substack{[s_m,t_m]\in\mathcal{M} \\ \tau\in\{s_m,\dots,t_m\}}} [\mathcal{L}(y_{s_m:\tau}) + \mathcal{L}(y_{\tau+1:t_m}) - \mathcal{L}(y_{s_m:t_m}) + \beta]$;
7 **if** $\mathcal{Q} < 0$ **then**
8 | $\hat{\tau} \leftarrow \operatorname{argmin}_{\substack{[s_m,t_m]\in\mathcal{M} \\ \tau\in\{s_m,\dots,t_m\}}} [\mathcal{L}(y_{s_m:\tau}) + \mathcal{L}(y_{\tau+1:t_m}) - \mathcal{L}(y_{s_m:t_m}) + \beta]$;
9 | $cp \leftarrow \{\hat{\tau},\ \text{WildBinSeg}(y_{s:\hat{\tau}}, \beta, M),\ \text{WildBinSeg}(y_{\hat{\tau}+1:t}, \beta, M)\}$;
10 | Return cp;
11 **end**
12 Return {};

that intervals of larger lengths are less frequent than intervals of smaller lengths; furthermore, the intervals are generated in such a way to cover uniformly the whole sequence. That is, our intervals set will be:

$$\mathcal{M} = \bigcup_{k=1}^{\lceil \log_{\frac{1}{a}}(n) \rceil} \bigcup_{i=1}^{m_k} \{[\lfloor (i-1)u_k \rfloor,\ \lceil (i-1)u_k + l_k \rceil]\}, \tag{2.8}$$

with $1/2 \leq a < 1$ being a decay parameter regulating the smallest segment length, $m_k = 2 * \lceil a^{1-k)} \rceil - 1$, $l_k = na^{k-1}$, $u_k = (n - l_k)/(m_k - 1)$. The new generating scheme falls exactly in place of the one in line 4 of Algorithm 5 with the rest of the procedure being the same. In this way it is possible to obtain some theoretical guarantees on the computational complexity of the method, which ends effectively being $\mathcal{O}(n \log(n))$, and on the asymptotic optimality of the procedure.

Narrowest Over Threshold. NOT, from Baranowski et al. (2016), is an alternative search scheme analogue to the BS schemes. Similarly to WBS it splits data into subsets and performs a test on each subset. But to then construct the set of estimated changepoints if follows the procedure:

1. keeps all segments whose test statistic is above some threshold;

2. orders these segments from shortest to longest;

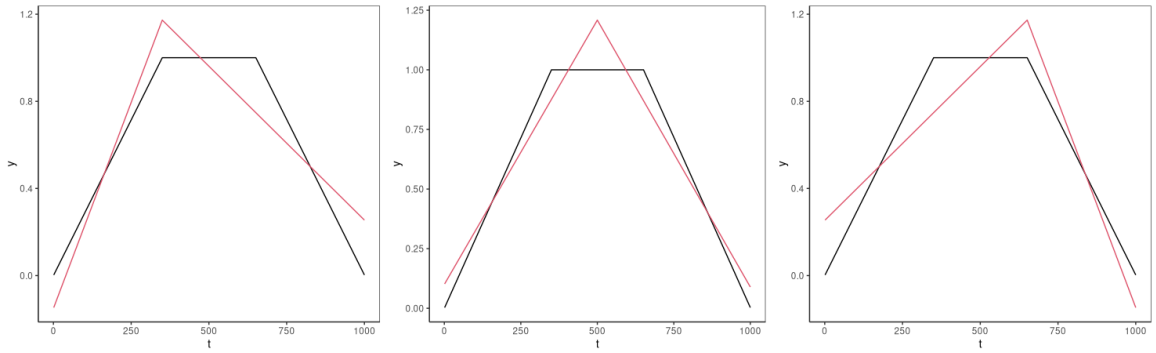


Figure 2.3: In black, the signal object of inference. In red, the best triangular approximation of such signal for a change at $\tau = 350$, on the left, $\tau = 500$, in the center, and $\tau = 750$, on the right.

3. adds changepoints by processing the list of segments following the new order; when each new changepoint is added later segments in the list that contain that changepoint are removed.

The idea is that all segments kept in the first stage show evidence for a changepoint. But we can most accurately estimate the position of a change if the segment only contains a single changepoint – and this is most likely to be for the shortest segments. Hence we process the segments from shortest to longest. As we do this we add the estimated change from the shortest segment, then remove all subsequent segments that contain the time at which that change occurred (as the signal in such segments may be because of the detected change), and then reiterate the procedure. To motivate NOT, let us consider the example in Figure 2.3. We observe a piecewise linear sequence of 100 observations with two true changes at 350 and 650. Let's say we wish to obtain the best $\ell 2$ triangular approximation of such signal – *i.e.* performing a single change estimation as in one iteration of BS. In such a scenario, if we had any BS scheme in place, such as WBS or SBS, we would flag a false positive. WBS would in fact estimate a change in the middle as the resulting segmentation is the one that achieves the smallest segmentation cost – minimizing the $\ell 2$ error. NOT avoids such error since it would only search through the narrowest interval, which is very likely to contain one single changepoint.

2.2.2 Extensions of Binary Segmentation

Possibly for their simplicity and ease of coding, many procedures rely on Binary Segmentation. We present some, by noting that what follows is certainly not a comprehensive list of such extensions.

Multivariate changepoint procedures. Several multivariate procedures take advantage of Binary Segmentation to extend to single change statistics to detect multiple changes. In Wang and Samworth (2018) a statistics for sparse change-in-mean is derived from the left singular vector of a CUSUM transformation on the time series matrix. Similarly in Cho and Fryzlewicz (2015) a BS method for detection of a change in the autocovariance and cross-covariance of a high-dimensional time series. In Leonardi and Bühlmann (2016) we find an efficient multivariate change-in-regression procedure.

Change in parameters of an ARCH model. In Fryzlewicz and Rao (2014) the BASTA procedure performs multiple changepoint detection in the piecewise constant parameters of an ARCH model. BASTA is a two step procedure consisting of an initial transformation of the series into a piecewise constant mean process $u_t = g(x_t, x_{t-1}, \ldots, x_{t-k})$ for given $g(\cdot)$ and k, followed by a regular Binary Segmentation approach. Multiple choices for $g(\cdot)$ are presented within the paper.

Non-stationary time-series. In Korkas and Fryzlewicz (2017) the WBS segmentation procedure is extended to deal with change detection in the second-order structure of a non-stationary time series. This is achieved through the decomposition of the series through local wavelet periodograms.

Nonparametric approaches. Recently, Ross (2021) presented a way to integrate nonparametric test statistics (like Mood, 1954) on a WBS scheme to build a multiple changepoint procedure. Lastly, the procedure presented in Chapter 5 can be essentially seen as an online nonparametric rolling window binary segmentation approach.

2.3 Other approaches

We conclude the chapter by mentioning other approaches to changepoint detection that do not directly fall in the super mentioned categories.

2.3.1 Sequential testing approaches

Here we present some offline sequential changepoint procedures based both on rolling window statistics or on cumulative sums. A more comprehensive review of such methodologies can be found in the background section of Chapter 4, which focus on the online changepoint detection problem, as many of those procedures find their roots in the sequential hypothesis testing literature. For a comprehensive review of the sequential procedures see Tartakovsky et al. (2014).

The MOSUM approach. Moving sums statistics originate as monitoring scheme procedures (Eiauer and Hackl, 1978; Chu et al., 1995) to be revisited recently as multiple changepoint detection procedures, see Eichinger and Kirch (2018). The idea

consists in a sequential evaluation of some test statistic in a rolling window, being a subset of the data that contains the most recent observations up to some iteration. The trace generated from the rolling window is then processed as a regular series in order to reconstruct the changepoint locations. More formally, let $y_{1:n} = y_1, \ldots, y_n$ be a realization from some stochastic process. For example, at time t we can construct the test statistics:

$$S_t = \max_{G \leq \tau \leq t-G} |S_{t,\tau}|, \tag{2.9}$$

$$S_{t,\tau} = \sum_{i=\tau+1}^{\tau+G} h(y_i) - \sum_{i=\tau-G+1}^{\tau} h(y_i). \tag{2.10}$$

Where $h(\cdot)$ is an estimator function of the parameter of interest θ, $G \in N$ is the bandwidth parameter that controls the size of the window, being $2G$. The statistic compares the difference of two sub-samples within the rolling window: a large difference is an indication of the presence of a change. Hence, intuitively, the MOSUM procedure maps a general changepoint problem to a change-in-mean problem. We compute the statistic for $t \in G, \ldots, n-G$ (where for $t < G$), obtaining its trace $S_{G:n-G}$. Then, to reconstruct the K changepoint estimates, for a given β, find $[l_k, u_k] : S_{l_k:u_k} > \beta \ \forall \ l_k \leq u_k < l_{k+1};\ k \in 1, \ldots, K$ the disjoint contiguous sets of the statistics that are over the threshold β. Within these sets, the value of t corresponding to the highest value of the statistics is picked as a changepoint. One of the most immediate advantages of MOSUM lies in the linear computational complexity of the procedure, being $\mathcal{O}(Gn)$, as only one pass over the entire sequence is necessary to reconstruct the trace of the statistics.

As an example, in Figure 2.4 we find an illustration of the MOSUM procedure on a Gaussian change-in-mean case. For this case, as reported in Eichinger and Kirch (2018), the full statistics becomes:

$$S_t = \max_{G \leq \tau \leq t-G} \frac{1}{\hat{\sigma}} \left| \frac{1}{\sqrt{2G}} \left(\sum_{i=\tau+1}^{\tau+G} y_i - \sum_{i=\tau-G+1}^{\tau} y_i \right) \right|,$$

where $\hat{\sigma}$ is an estimate of the long-run variance necessary to ensure that the behaviour of the partial sums is not affected by the number of changepoints. Choices for the bandwidth parameter G, the threshold β and a long-run variance estimator for σ, as well as asymptotic guarantees on the power of the test are provided within the Eichinger and Kirch (2018).

For a general description of the MOSUM procedure and a comprehensive review of its extensions, please refer to Reckrühm (2019).

Sequential Likelihood Ratio tests. Dette and Gösmann (2020) present a sequential likelihood ratio test procedure to test for a single change in the parameters

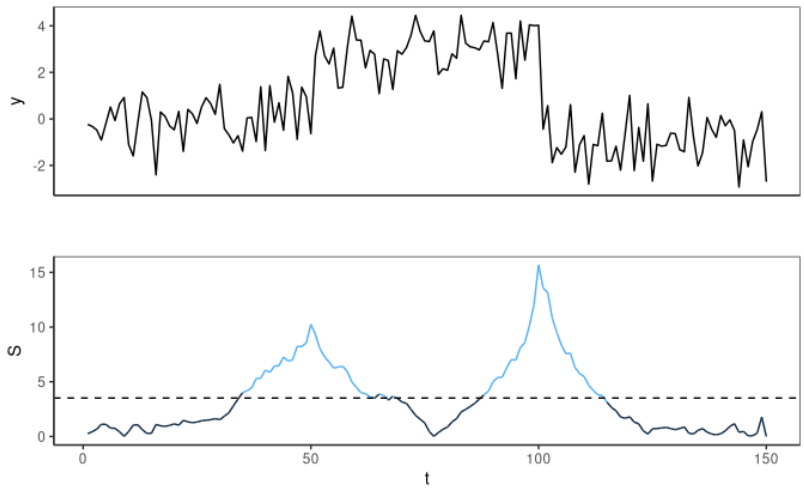


Figure 2.4: On top, a realization of a simple Gaussian change-in-mean scenario with changes at 50 and 100. At the bottom, the value of the MOSUM statistics relative to each time point. The dotted line corresponds to a threshold of 3.5, with the values over the threshold highlighted in blue: in this case MOSUM correctly identifies the two changepoints.

of some distribution. The procedure is based on cumulative sums of an estimator. That is, we observe $y_t \sim f(\theta_t)$ for $t = 1, \dots, n$, and we wish to sequentially test for:

$$H_0 : \theta_1 = \cdots = \theta_m = \dots,$$

against

$$H_1 : \theta_1 = \cdots = \theta_m = \cdots = \theta_\tau \neq \theta_{\tau+1} = \theta_{\tau+2} = \dots$$

for $m > 1$ where $y_{1:m}$ is stable under the parameters of interest and $\tau \geq m$ is a changepoint. The sequential test statistics, up to time t, will be given by:

$$S_t = \max_{1 \leq \tau < t} \left(\sum_{i=\tau+1}^{t} h(y_i) - \sum_{i=1}^{\tau} h(y_i) \right)^2,$$

where $h(\cdot)$ is an estimator function of the parameter of interest θ. Reconstructing the trace of such statistics up to time n has a computational complexity of $\mathcal{O}(n^2)$, as for computing the partial sums each iteration is linear in n. This issue is addressed in 4 where the computational complexity is of particular interest. In Dette and Gösmann (2020) several test statistics are introduced to cover a more general class of parameters and multivariate applications, amongst those the MOSUM statistics is derived as a special case essentially obtaining an equivalent formulation to what already presented in 2.9.

2.3.2 Penalised approaches (fused lasso)

The idea behind lasso optimization approaches is to express the changepoint problem as a $\ell 1$ penalised convex optimization problem. This approach is relatively close to the penalised optimization problem formulated in 2.4, where the difference between the two optimization problems is essentially in the type of penalty applied to the minimization. Originating from Harchaoui and Lévy-Leduc (2007), focusing on the simple Gaussian change-in-mean case, the Cachalot[1] procedure solves:

$$\min_{\mu_{1:n}} \sum_{t=1}^{n} (y_t - \mu_t)^2 + \kappa \sum_{t=2}^{n} |\mu_t - \mu_{t-1}| \tag{2.11}$$

with $\kappa > 0$ being an $\ell 1$ penalty on the magnitude of a change. A combined LARS/LASSO and dynamical programming approach is then employed in order to solve the optimization efficiently in $\mathcal{O}(n^2)$.

A generalization to the multivariate case is derived in Bleakley and Vert (2011). In Kim et al. (2009); Tibshirani et al. (2014), a similar optimization problem is solved, with a penalty on the discrete derivative of the mean signal: as we are going to cover this will be particularly helpful in case of a fluctuating mean signal. Similarly, as a comparison to our $\ell 2 + \ell 0$ optimization method introduced in Chapter 3, in Section 3.6.3 we frame the LAVA $\ell 2 + \ell 1$ optimization problem (from Chernozhukov et al., 2017) as a changepoint optimization problem.

2.3.3 Model Based approaches

The last approaches we review are based on formulating a model or log-likelihood of the data with changes and estimate the resulting parameters through either a Bayesian approach or EM approach.

Bayesian approaches. Historically, changepoint detection has always been of interest to Bayesian Analysis with numerous works from the field (amongst those we mention Raftery and Akman, 1986; Barry and Hartigan, 1993; Liu and Lawrence, 1999). Usually, the idea is to specify a model that allows for one or more changes and to estimate the resulting parameters through the Bayesian inferential approach. A prior can be placed either on the within-change parameters, the changepoint locations or a point process regulating the distance between two successive changepoints. To obtain inference from the posterior the sampling can be either done exactly (for example we mention Fearnhead, 2006) or through MCMC (Stephens, 1994) or reversible jump MCMC approaches (as in Green, 1995). The Bayesian approach to changepoint detection differs from those introduced so far both in terms of the

[1] With possibly one of the most well thought acronyms in the literature, for CAtching CHAngepoints with LassO.

initial parameters and of the final results: a Bayesian methodology relies on prior specifications and can therefore be quite sensitive to such choices; a richer output is produced that can more easily quantify uncertainty about changes, but it can be harder to present a simple summary of the estimates (see Siems et al., 2019).

Mixed Linear Modelling EM approaches. Picard et al. (2011) introduce a procedure for performing multivariate changepoint detection through a blended approach of Expectation-Maximization and Dynamical Programming. The procedure relies on a mixed linear model with covariates and an additional parameter for expressing some piecewise-constant signal. An EM procedure, namely the ECM algorithm (Meng and Rubin, 1993), is then employed to estimate all the model parameters. In this procedure the M step is broken into two sub-steps: an optimization step for the parameters not subject to the changes, and a constrained dynamical programming step for the optimization of the piecewise-constant parameters across the multiple series.

HMM EM approaches. In this paradigm we wish to reconstruct the state of a discrete Hidden Markov Model starting from the observations, assuming that each y_i is a realization of one of the K^{th} different models $f_k(y_i),\ k = 1, \ldots, K$.

As an example, we present the Expectation Maximization Change Point (EMCP) methodology from Chang and Lu (2016), a multiple segments multivariate changepoint procedure. They lay the complete log-likelihood function of the data, for $t = 1, \ldots, n$ observations, $j = 1, \ldots, m$ variates:

$$\sum_{k=1}^{K} \sum_{t=1}^{n} z_{k,t} \sum_{j=1}^{m} \log f_k(y_{t,j}, \theta_k)$$

where $f_k(\cdot)$ is the likelihood function for the data from group k – for instance, in the Gaussian change-in-mean and variance $f_k(y) = \frac{1}{\sigma_k \sqrt{2\pi}} \exp\left[-(y - \mu_k)/2\sigma_k^2\right]$ – and the $z_{kt} \in \{0, 1\}$ is a Bernoulli random variable denoting belonging of a given observation to a specific group and is defined as following:

$$z_{k,t} = \begin{cases} 1, & \text{if the } t^{th}\text{observation belongs to model } k, \\ 0, & \text{otherwise.} \end{cases}$$

Estimations of both the $z_{k,1:n}$ variables, which are treated as unobserved data, as well as model parameters θ_k is performed then through an iterative Expectation-Maximization approach.

Chapter 3

Detecting Changes in Autocorrelated and Fluctuating Signals

3.1 Introduction

Detecting changes in data streams is a ubiquitous challenge across many modern applications of statistics. It is important in such diverse areas as bioinformatics (Olshen et al., 2004; Futschik et al., 2014), ion channels (Hotz et al., 2013), climate records (Reeves et al., 2007), oceanographic data (Killick et al., 2010) and finance (Kim et al., 2005). The most common and important change detection problem is that of detecting changes in mean, and there have been a large number of different approaches to this problem that have been proposed (e.g. Olshen et al., 2004; Killick et al., 2012; Fryzlewicz, 2014; Frick et al., 2014; Maidstone et al., 2017; Eichinger and Kirch, 2018; Fearnhead and Rigaill, 2019; Fryzlewicz, 2018, amongst many others). Almost all of these methods are based on modelling the data as having a constant mean between changes and the noise in the data being independent. Furthermore, all changepoint methods require specifying some threshold or penalty that affects the amount of evidence that there needs to be for a change before an additional changepoint is detected. In general the methods have default choices of these thresholds or penalties that have good theoretical properties under strong modelling assumptions.

Whilst these methods perform well when analysing simulated data where the assumptions of the method hold, they can be less reliable in real applications, particularly if the default threshold or penalties are used. Reasons for this include the noise in the data being autocorrelated, or the underlying mean fluctuating slightly between the abrupt changes that one wishes to detect. To see this, consider change

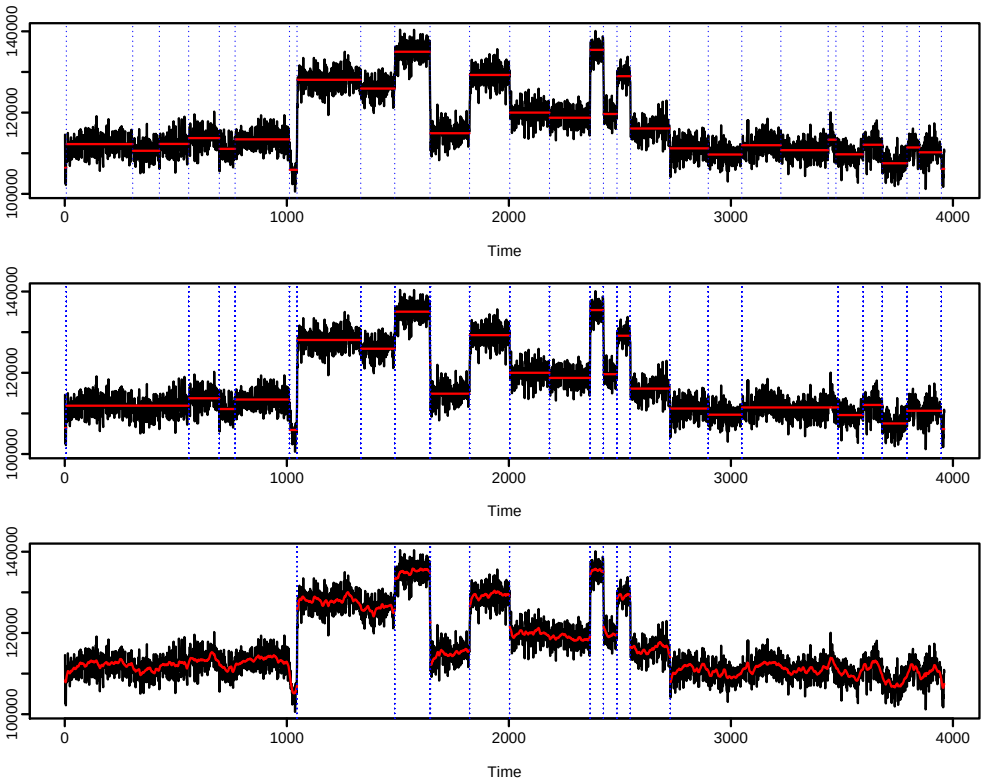


Figure 3.1: Segmentations of well-log data: wild binary segmentation using the strengthened Schwarz information criteria (top); segmentation under square error loss with penalty inflated to account for autocorrelation in measurement error (middle); optimal segmentation from DeCAFS with default penalty (bottom). Each plot shows the data (black line) the estimated mean (red line) and changepoint location (vertical blue dashed lines).

detection for the well-log data (taken from Ruanaidh and Fitzgerald, 2012; Fearnhead and Liu, 2011) shown in Figure 3.1. This data comes from lowering a probe into a bore-hole, and taking measurements of the rock structure as the probe is lowered. The data we plot has had outliers removed. As the probe moves from one rock strata to another we expect to see an abrupt change in the signal from the measurements, and it is these changes that an analyst would wish to detect. Previous analyses of this data have shown that, marginally, the noise in the data is very well approximated by a Gaussian distribution; but by eye we can see local fluctuations in the data that suggest either autocorrelation in the measurement error, or structure in the mean between the abrupt changes.

The top plot shows an analysis of the well-log data that uses wild binary segmentation (Fryzlewicz, 2014) with the standard cusum test for a change in

mean, and then estimates the number of changepoints based on a strengthened Schwarz information criteria. Both the cusum test and the strengthened Schwarz information criteria are based on modelling assumptions of a constant mean between changepoints and independent, identically-distributed (IID) Gaussian noise, and are known to consistently estimate the number and location of the changepoints if these assumptions are correct. However in this case we can see that it massively overfits the number of changepoints. Similar results are obtained for standard implementation of other algorithms for detecting changes in mean, see Figure 13 in the Supplementary Material.

Lavielle and Moulines (2000) and Bardwell et al. (2019) suggest that if we estimate changepoints by minimising the squared error loss of our fit with a penalty for each change, then we can correct for potential autocorrelation in the noise by inflating the penalty used for adding a changepoint. The middle plot of Figure 3.1 shows results for such an approach (Bardwell et al., 2019); this gives an improved result but it still noticeably overfits.

By comparison, the method we propose models both autocorrelation in the noise and local fluctuations in the mean between changepoints – and analysis of the data using default settings produces a much more reasonable segmentation of the data (see bottom plot of Figure 3.1). This method is model-based, and assumes that the local fluctuations in the mean are realisations of a random walk and that the noise process is an AR(1) process. We then segment the data by minimising a penalised cost that is based on the log-likelihood of our model together with a BIC penalty for adding a changepoint.

The key algorithmic challenge with our approach is minimising the penalised cost. In particular many existing dynamic programming approaches (e.g. Jackson et al., 2005; Killick et al., 2012) do not work for our problem due to the dependence across segments caused by the autocorrelated noise. We introduce a novel extension of the functional pruned optimal partitioning algorithm of Maidstone et al. (2017), and we call the resulting algorithm DeCAFS, for Detecting Changes in Autocorrelated and Fluctuating Signals. It is both computationally efficient (analysis of the approx 4000 data points in the well-log data taking a fraction of a second on a standard laptop) and guaranteed to find the best segmentation under our criteria.

Whilst we are unaware of any previous method that tries to model both autocorrelation and local fluctuations, Chakar et al. (2017) introduced AR1Seg which aims to detect changes in mean in the presence of autocorrelation. Their approach is similar to ours if we remove the random walk component, as they aim to minimise a penalised cost where the cost is the negative of the log-likelihood under a model with an AR(1) noise process. However they were unable to minimise this penalised cost, and instead minimised an approximation that removes the dependence across segments. One consequence of using this approximation is that it often estimates two

consecutive changes at each changepoint, and AR1Seg uses a further post-processing step to try and correct this. Moreover, our simulation results show that using the approximation leads to a loss of power, particularly when the autocorrelation in the noise is high.

Particularly if interest lies in estimating how the underlying mean of the data varies over time, natural alternatives to DeCAFS are trend filtering methods (Kim et al., 2009; Tibshirani et al., 2014) which estimate the mean function under an L_1 penalty on a suitably chosen discrete derivative of the mean. Depending on the order of the derivative, these methods can fit a piecewise constant (in this case trend filtering is equivalent to the fused lasso of Tibshirani et al., 2005), linear, or quadratic etc. function to the mean. One advantage of trend filtering is its flexibility: depending on the application one can fit mean functions with different degrees of smoothness. However it does not allow one to jointly estimate a smoothly changing mean and abrupt changes – the L_1 penalty must be chosen to capture one or other of these effects. This is particularly an issue if the main interest is in detecting abrupt changes rather than estimating the mean. These issues are investigated empirically in Section F of the Supplementary Material.

Distinguishing between local fluctuations and abrupt changes is possible by methods that model the mean as a sum of functions (Jalali et al., 2013) – for example one smoothly varying and one piecewise constant. And trend-filtering, or other regularised estimators can then be used to estimate each component. The LAVA approach of Chernozhukov et al. (2017) is one such approach, fitting the piecewise constant function and the smoothly varying function using, respectively, an L_1 and L_2 penalty on the function's first discrete derivative. The main difference between LAVA and DeCAFS is thus that LAVA has an L_1 penalty for abrupt changes, and DeCAFS has an L_0 penalty. If interest is primarily in detecting changes, previous work has shown the use of an L_0 penalty to be preferable to an L_1 penalty – as the latter can often over-fit the number of changes (see e.g. the empirical evidence in Fearnhead et al., 2018; Jewell et al., 2020), and a post-processing step is often needed to correct for this (Lin et al., 2017; Safikhani and Shojaie, 2020).

One important feature of modelling the random fluctuations in the mean via a random walk, is that the information that a data point y_s has about a change at time t decays to 0 as $|t - s|$ gets large. This is most easily seen if we consider applying DeCAFS to detect a single abrupt change. We show in Section 3.5 that whether DeCAFS detects a change at a time t depends on some contrast of the data before and after t. This contrast compares a weighted mean of the data before t to a weighted mean of the data after t, with the weights decaying (essentially) geometrically with the length of time before/after t. This is appropriate for the random walk model which enables e.g. the mean of y_{t+h} to be very different to the mean at y_{t+1} if $h > 0$ is large, and thus y_{t+h} contains little to no information about whether there has been

an abrupt change in the mean of y_{t+1} compared to the mean of y_t. By contrast, standard CUSUM methods for detecting a change in mean quantify the evidence for a change at t by a contrast of the *unweighted mean* of the data before and after t – thus each data point, y_{t+h}, has the same amount of information about the change regardless of the value of h. In situations where there are local fluctuations in the mean but through some stationary process, such as a mean-reverting random walk, data far from a change would still have a non-negligble amount of information about it. In such situations, particularly if the segments are long, DeCAFS could have lower power than CUSUM or other methods that ignore any local fluctuations in the data. We investigate this empirically in Section A.4.2.

The outline of the paper is as follows. In the next section we introduce our model-based approach and the associated penalised cost. In Section 3.3 we present DeCAFS, a novel dynamic programming algorithm that can exactly minimise the penalised cost. To implement our method we need estimates of the model parameters, and we present a simple way of pre-processing the data to obtain these in Section 3.4. We then look at the theoretical properties of the method. These justify the use of the BIC penalty, show that our method has more power at detecting changes when our model assumptions are correct than standard approaches, and also that we have some robustness to model error – in that we can still consistently estimate the number and location of the changepoints in such cases by adapting the penalty for adding a changepoint. Sections 3.6 and 3.7 evaluate the new method on simulated and real data; and the paper ends with a discussion.

Code implementing the new algorithm is available in the R package `DeCAFS` on CRAN. The package and full code from our simulation study is also available at `github.com/gtromano/DeCAFS`.

3.2 Modelling and Detecting Abrupt Changes

3.2.1 Model

Let $y_{1:n} = (y_1, \ldots, y_n) \in \mathbb{R}^n$ be a sequence of n observations, and assume we wish to detect abrupt changes in the mean of this data in the presence of local fluctuations and autocorrelated noise. We take a model-based approach where the signal vector is a realisation of a random walk process with abrupt changes, and we super-impose an AR(1) noise process.

So for $t = 1, \ldots, n$,

$$y_t = \mu_t + \epsilon_t, \tag{3.1}$$

where for $t = 2, \ldots, n$

$$\mu_t = \mu_{t-1} + \eta_t + \delta_t, \quad \text{with } \eta_t \underset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\eta^2),\ \delta_t \in \mathbb{R}, \tag{3.2}$$

and $\delta_t = 0$ except at time points immediately after a set of m changepoints, $0 < \tau_1 < \cdots < \tau_m < n$. That is $\delta_t = 0$ unless $t = \tau_j + 1$ for some j. This model is unidentifiable at changepoints. If τ is a changepoint, then whilst the data is informative about $\mu_{\tau+1}$ and μ_τ, we have no further information about the specific value of $\delta_{\tau+1}$ relative to $\eta_{\tau+1}$. We thus take the convention that $\delta_{\tau+1} = \mu_{\tau+1} - \mu_\tau$ and $\eta_{\tau+1} = 0$, which is the most likely value of $\eta_{\tau+1}$ under our model, and consistent with maximising the likelihood criteria we introduce below. The noise process, ϵ_t is a stationary AR(1) process with, for $t = 2, \ldots, n$,

$$\epsilon_t = \phi\epsilon_{t-1} + \nu_t \quad \text{with } \nu_t \underset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\nu^2), \tag{3.3}$$

for some autocorrelation parameter, ϕ with $|\phi| < 1$ and $\epsilon_1 \sim \mathcal{N}(0, \sigma_\nu^2/(1-\phi^2))$.

Special cases of our model occur when $\phi = 0$ or when $\sigma_\eta^2 = 0$. When $\phi = 0$ our noise process ϵ_t is then IID, and the model is equivalent to a random walk plus noise with abrupt changes. When $\sigma_\eta^2 = 0$ we are detecting changes in mean with an AR(1) noise process, resulting in a formulation equivalent to the one of Chakar et al. (2017).

3.2.2 Penalised Maximum Likelihood Approach

In the following we will assume that ϕ, σ_η^2 and σ_ν^2 are known; we consider robust approaches to estimate these parameters from the data in Section 3.4. We can then write down a type of likelihood for our model, defined as the joint density of the observations, $y_{1:n}$, and the local fluctuations in the mean, $\eta_{2:n}$. We will express this as a function of $\mu_{1:n}$ and $\delta_{2:n}$. Writing $f(\cdot|\cdot)$ for a generic conditional density, we have that this is

$$\begin{aligned}
\mathcal{L}(y_{1:n}; \mu_{1:n}, \delta_{2:n}) &= \left(\prod_{t=2}^{n} f(\mu_t|\mu_{t-1}, \delta_t)\right) f(y_1|\mu_1) \left(\prod_{t=2}^{n} f(y_t|y_{t-1}, \mu_{t-1}, \mu_t)\right) \\
&\propto \left(\prod_{t=2}^{n} \exp\left\{-\frac{(\mu_t - \mu_{t-1} - \delta_t)^2}{2\sigma_\eta^2}\right\}\right) \exp\left\{-\frac{(y_1 - \mu_1)^2}{2\sigma_\nu^2/(1-\phi^2)}\right\} \\
&\quad \times \left(\prod_{t=2}^{n} \exp\left\{-\frac{((y_t - \mu_t) - \phi(y_{t-1} - \mu_{t-1}))^2}{2\sigma_\nu^2}\right\}\right).
\end{aligned}$$

We have used the specific Gaussian densities of our model, and dropped multiplicative constants, to get the second expression.

If we knew the number of changepoints we could estimate their position by maximising this likelihood subject to the constraints on the number of non-zero entries of $\delta_{2:n}$. However, as we need to also estimate the number of changepoints we proceed by maximising a penalised version of the log of the likelihood where we introduce a penalty for each changepoint – this is a common approach to changepoint detection, see e.g. Maidstone et al. (2017). It is customary to restate this as minimising a penalised cost, rather than maximising a penalised likelihood, where the cost is minus twice the log-likelihood. That is we estimate the number and location of the changepoints by solving the following minimisation problem:

$$\mathcal{Q}_n = \min_{\substack{\mu_{1:n} \\ \delta_{2:n}}} \Big\{ (1-\phi^2)\gamma(y_1-\mu_1)^2 + \sum_{t=2}^{n} \Big[\lambda(\mu_t - \mu_{t-1} - \delta_t)^2 + \gamma\Big((y_t - \mu_t) - \phi(y_{t-1} - \mu_{t-1}) \Big)^2 + \beta \, \mathbb{1}_{\delta_t \neq 0} \Big] \Big\}, \quad (3.4)$$

where $\beta > 0$ is the penalty for adding a changepoint, $\lambda = 1/\sigma_\eta^2$, $\gamma = 1/\sigma_\nu^2$, and $\mathbb{1} \in \{0, 1\}$ is an indicator function. For the special case of a constant mean between changepoints, corresponding to $\sigma_\eta^2 = 0$, we require $\mu_t = \mu_{t-1} + \delta_t \; \forall \; t = 2, \ldots, n$ and simply drop the first term in the sum.

3.2.3 Dynamic Programming Recursion

We will use dynamic programming to minimise the penalised cost (3.4). The challenge here is to deal with the dependence across changepoints due to the AR(1) noise process which means that some standard dynamic approaches for changepoint detection, such as optimal partitioning (Jackson et al., 2005) and PELT (Killick et al., 2012), cannot be used. To overcome this, as in Rigaill (2015) or Maidstone et al. (2017), we define the function $\mu \mapsto Q_t(\mu)$ to be the minimum penalised cost for data $y_{1:t}$ conditional on $\mu_t = \mu$,

$$Q_t(\mu) = \min_{\substack{\mu_{1:t} \\ \delta_{2:t}, \mu_t = \mu}} \Big\{ (1-\phi^2)\gamma(y_1-\mu_1)^2 + \sum_{i=2}^{t} \Big[\lambda(\mu_i - \mu_{i-1} - \delta_i)^2 + \gamma\Big((y_i - \mu_i) - \phi(y_{i-1} - \mu_{i-1}) \Big)^2 + \beta \, \mathbb{1}_{\delta_i \neq 0} \Big] \Big\}.$$

So $\mathcal{Q}_n = \min_{\mu \in \mathbb{R}} Q_n(\mu)$; and the following proposition gives a recursion for $Q_t(\mu)$.

Proposition 1 *The set of functions* $\{\mu \mapsto Q_t(\mu)\,,\, t = 1, \ldots, n\}$ *satisfies*

$$Q_1(\mu) = (1-\phi^2)\gamma(y_1-\mu)^2 \text{ and, for } t = 2,\ldots,n,$$

$$Q_t(\mu) = \min_{u\in\mathbb{R}}\left\{Q_{t-1}(u) + \min\{\lambda(\mu-u)^2,\beta\} + \gamma\Big((y_t-\mu)-\phi(y_{t-1}-u)\Big)^2\right\}. \quad (3.5)$$

The intuition behind the recursion is that we first condition on $\mu_{t-1} = u$, with the term in braces being the minimum penalised cost for $y_{1:t}$ given u and $\mu_t = \mu$, and then minimise over u. The cost in braces is the sum of three terms: (i) the minimum penalised cost for $y_{1:t-1}$ given u; (ii) the cost for the change in mean from u to μ; and (iii) the cost of fitting data point y_t with μ_t. The cost for the change in mean, (ii), is just the minimum of the constant cost for adding a change and the quadratic cost for a change due to the random walk. The recursion applies to the special case of a constant mean between changepoints, where $\lambda = \infty$, if we replace $\min\{\lambda(\mu-u)^2,\beta\}$ with its limit as $\lambda\to\infty$, which is $\beta\mathbb{1}_{\mu\neq u}$.

Whilst recursions of this form have been considered in earlier changepoint algorithms (e.g. Maidstone et al., 2017; Hocking et al., 2020), the dependence between the current mean u and the previous mean μ that appears in terms (ii) and (iii) makes our recursion more challenging to solve. We next show one efficient way of solving by combining existing functional pruning dynamic programming ideas with properties of infimal convolutions.

3.3 Computationally Efficient Algorithm

3.3.1 The DeCAFS Algorithm

Algorithm 6 gives pseudo code for solving the dynamic programming recursion introduced in Proposition 1. The key to implementing this algorithm is performing the calculations in line 5, and how this can be done efficiently will be described below. Throughout we give the algorithm for the case where there is a random walk component, i.e. $\lambda < \infty$, though it is trivial to adapt the algorithm to the $\lambda = \infty$ case.

As well as solving the recursion for $Q_t(\mu)$, Algorithm 6 shows how we can also obtain the estimate of the mean, through a standard back-tracking step. The idea is that our estimate of μ_n, $\hat{\mu}_n$, is just the value of μ that maximises $Q_n(\mu)$. We then loop backwards through the data, and our estimate of μ_t is the value that minimises the penalised cost for the data $y_{1:t}$ conditional on $\mu_{t+1} = \hat{\mu}_{t+1}$, which can be calculated as $B_t(\mu)$ in line 11.

Finally, as we obtain the estimates of the mean, we can also directly obtain the estimated changepoint locations. It is straightforward to see, by examining the form of the penalised cost, that the optimal solution for $\delta_{2:n}$ has $\delta_{t+1} \neq 0$ (and hence t is a changepoint) if and only if $\lambda(\hat{\mu}_{t+1}-\hat{\mu}_t)^2 > \beta$.

Algorithm 6: DeCAFS

Data: $\mathbf{y} = y_{1:n}$ a time series of length n
Input: $\beta > 0$, $\lambda > 0$, $\gamma > 0$ and $0 \leq \phi < 1$.

begin Initialisation
  $Q_1(\mu) \leftarrow (1-\phi^2)\gamma(y_1 - \mu)^2$
end
for $t = 2$ *to* n **do**
  $Q_t(\mu) \leftarrow$
  $$\min_u \left\{ Q_{t-1}(u) + \min\{\lambda(\mu - u)^2, \beta\} + \gamma\Big((y_t - \mu) - \phi(y_{t-1} - u)\Big)^2 \right\}$$
end
begin Backtracking
  $\hat{\mu}_n \leftarrow \operatorname{argmin} Q_n(\mu)$
  $\hat{\tau} \leftarrow n$
  for $t = n-1$ *to* 1 **do**
    $B_t(\mu) \leftarrow Q_t(\mu) + \min\{\lambda(\mu - \hat{\mu}_{t+1})^2, \beta\} + \gamma\Big((y_{t+1} - \hat{\mu}_{t+1}) - \phi(y_t - \mu)\Big)^2$
    $\hat{\mu}_t \leftarrow \operatorname{argmin} B_t(\mu)$
    if $(\hat{\mu}_t - \hat{\mu}_{t+1})^2 > \beta/\lambda$ **then**
      $\hat{\tau} \leftarrow (t, \hat{\tau})$
    end
  end
end
Return $\hat{\mu}_{1:n}$, $\hat{\tau}$

3.3.2 The Infimal Convolution

The main challenge with Algorithm 6 is implementing line 5. Firstly this needs a compact way of characterising $Q_t(\mu)$. This is possible as $Q_1(\mu)$ is a quadratic function; and the recursion maps piecewise quadratic functions to piecewise quadratic functions. Hence $Q_t(\mu)$ will be piecewise quadratic and can be defined by storing a partition of the real-line together with the coefficients of the quadratics for each interval in this partition.

Next we can simplify line 5 of Algorithm 6. As written line 5 involves minimising a two-dimensional function, in $(u, \mu) \in \mathbb{R}^2$, over the variable u. We can recast this operation into a one-dimensional problem by introducing the concept of an infimal convolution (see Chapter 12 of Bauschke and Combettes, 2011).

Definition 1 *Let f be a real-valued function defined on $\mathbb{R}$ and ω a non-negative scalar. We define $\mathsf{INF}_{f,\infty}(\theta) = f(\theta)$ and for $\omega > 0$,*

$$\mathsf{INF}_{f,\omega}(\theta) = \min_{u \in \mathbb{R}} \left(f(u) + \omega(u - \theta)^2 \right), \tag{3.6}$$

as the infimal convolution of f with a quadratic term.

The following proposition presents a reformulation of the update-rule into a minimization involving infimal convolutions, for the case $\phi \geq 0$. The proof is in Section A.2, together with details of equivalent results when $\phi < 0$.

Proposition 2 *Assume $\phi \geq 0$. The functions $\{Q_t(\mu)\,,\, t = 2, \ldots, n\}$ can be written as*

$$Q_t(\mu) = \min \left\{ Q_t^{=}(\mu),\ Q_t^{\neq}(\mu) \right\},$$

where

$$Q_t^{=}(\mu) = \mathsf{INF}_{\mathbb{Q}_{t-1}, \gamma\phi + \lambda}(\mu) + \tfrac{\gamma}{1-\phi} \Big(y_t - \phi y_{t-1} - (1-\phi)\mu \Big)^2,$$
$$Q_t^{\neq}(\mu) = \mathsf{INF}_{\mathbb{Q}_{t-1}, \gamma\phi}(\mu) + \tfrac{\gamma}{1-\phi} \Big(y_t - \phi y_{t-1} - (1-\phi)\mu \Big)^2 + \beta\,,$$

and

$$\mathbb{Q}_{t-1}(u) = Q_{t-1}(u) - \gamma\phi(1-\phi) \left(u - \frac{y_t - \phi y_{t-1}}{1-\phi} \right)^2.$$

3.3.3 Fast Infimal Convolution Computation

As noted above we can represent Q_t by $\mathsf{Q}_t = (q_t^1, ..., q_t^s)$ where each q_t^i is a quadratic defined on some interval $[d_i, d_{i+1}[$ with $d_1 = -\infty$ and $d_{s+1} = +\infty$. It is this representation of Q_t that we update at each time step. Some operations involved in solving the recursion, such as adding a quadratic to a piecewise quadratic, or

calculating the pointwise minimum of two piecewise quadratics are easy to perform with a computational cost that is linear in the number of intervals (see e.g. Rigaill, 2015). The following theorem shows that a fast update for the infimal convolution of a piecewise quadratic is also possible, and is important for developing a fast algorithm for solving the dynamic programming recursions.

Theorem 1 *Let $\mathsf{Q}_t = (q_t^1, ..., q_t^s)$ be the representation of the functional cost Q_t. For all $\omega \geq 0$, the representation returned by the infimal convolution $\mathsf{INF}_{Q_t,\omega}$ has the following order-preserving form:*

$$\mathsf{INF}_{\mathsf{Q}_t,\omega} = (\mathsf{INF}q_t^{u_1}, \mathsf{INF}q_t^{u_2}, ..., \mathsf{INF}q_t^{u_{s^*-1}}, \mathsf{INF}q_t^{u_{s^*}}),$$

with $1 = u_1 < u_2 < ... < u_{s^*-1} < u_{s^*} = s$ and $s^* \leq s$.

The key part of this result is that the order of the quadratics is not changed when we apply the infimal convolution, and thus we can calculate $\mathsf{INF}_{Q_t,\omega}$ using a linear scan over the real-line. The proof of this theorem is given in Appendix C of the supplementary materials of Romano et al. (2021), and an example algorithm for calculating $\mathsf{INF}_{Q_t,\omega}$ with complexity that is linear in s is shown in Section A.3.

3.4 Robust Parameter Estimation

Our optimisation problem (3.4) depends on three unknown parameters: σ_η^2, σ_ν^2 and ϕ. We estimate these parameters by fitting to robust estimates of the variance of the k-lag differenced data, $z_t^k = y_{t+k} - y_t$, for $k \geq 1$.

Proposition 3 *With the model defined by (3.1) – (3.3),*

$$z_t^k \sim \mathcal{N}\Big(\sum_{i=t+1}^{t+k} \delta_i, k\sigma_\eta^2 + 2\frac{1-\phi^k}{1-\phi^2}\sigma_\nu^2 \Big), \quad t = 1, \ldots, n-k.$$

Providing k is small relative to the average length of a segment, the mean of z_t^k will be zero for most t: if there are m changes then at most km of the z_t^ks will overlap a change and have a non-zero mean. A way to alleviate this issue is to estimate the variance of z_t^k through a robust estimator, such as the median absolute difference from the median, or MAD, estimator.

Fix K, and let v_k be the MAD estimator of the variance of z_t^k for $k = 1, \ldots, K$. We estimate the parameters by minimising the least square fit to these estimates,

$$\mathcal{S}_\phi(\sigma_\eta^2, \sigma_\nu^2) = \sum_{k=1}^{K} \Big(k\sigma_\eta^2 + 2\frac{1-\phi^k}{1-\phi^2}\sigma_\nu^2 - v_k \Big)^2.$$

In practice we can minimise this criteria by using a grid of values for ϕ and then for each ϕ value analytically minimise with respect to $\sigma_\eta^2 \geq 0$ and $\sigma_\nu^2 \geq 0$. Obviously, if we are fitting a model without the random walk component we can set $\sigma_\eta^2 = 0$, or if we wish to have uncorrelated noise we set $\phi = 0$. A remark: the least square estimate implicitly assumes that the z_t^ks are independent. Such an assumption does not hold on our model, however, ignoring the dependence in the z_t^ks would just slightly reduce the statistical efficiency of the estimator.

An empirical evaluation of this method for estimating the parameters is shown in Appendix F.1 of the supplementary materials of Romano et al. (2021). These include an investigation of the accuracy in situations where we have changepoints. In our simulation study we use $K = 10$, though similar results were obtained as we varied K.

3.5 Theoretical Properties

We can reformulate our model as a linear-regression. To do this it is helpful to introduce new variables, $\tilde{\eta}_{1:n}$, that give the cumulative effect of the random-walk fluctuations. To simplify exposition it is further helpful to define this process so it has an invertible covariance matrix. So we will let $\tilde{\eta}_1 \sim \mathcal{N}(0, \sigma_\eta^2)$ and $\tilde{\eta}_t = \tilde{\eta}_{t-1} + \eta_t$ for $t = 2, \ldots, n$. For a set of m changepoints $\tau_{1:m}$, and defining $\tau_0 = 0$, we can introduce a $n \times (m+1)$ matrix $X_{\tau_{0:m}}$ where the ith column is a column of τ_{i-1} zeros followed by $n - \tau_{i-1}$ ones. Our model is then

$$y_{1:n} = X_{\tau_{0:m}} \Delta + \zeta_{1:n}, \tag{3.7}$$

where $\zeta_{1:n}$ is a vector of Gaussian random variables with

$$\mathrm{Var}(\zeta_{1:n}) = \mathrm{Var}(\epsilon_{1:n}) + \mathrm{Var}(\tilde{\eta}_{1:n}) := \Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}}$$

the sum of the variance matrices for the AR component of the model, $\epsilon_{1:n}$, and the random walk component of the model, $\tilde{\eta}_{1:n}$; and Δ is a $(m+1) \times 1$ vector whose first entry is $\mu_1 - \tilde{\eta}_1$ and whose ith entry is $\delta_{\tau_{i-1}+1}$ the change at the $(i-1)$th changepoint. This formulation also allows us to consider the impact of model error on DeCAFS. Later, when we consider its asymptotic properties, we will allow for the data generating process to be (3.7) but with $\mathrm{Var}(\zeta_{1:n})$ different from that assumed by DeCAFS.

As shown in Section E of the Supplementary Material of Romano et al. (2021), the unpenalised version of the cost that we minimise, conditional on a specific set of changepoints, can be written as

$$\mathcal{C}(\tau_{1:m}) = \min_{\Delta, \tilde{\eta}_{1:n}, \tilde{\eta}_1 = 0} \left[(y_{1:n} - X_{\tau_{0:m}} \Delta - \tilde{\eta}_{1:n})^T \Sigma_{\mathsf{AR}}^{-1} (y_{1:n} - X_{\tau_{0:m}} \Delta - \tilde{\eta}_{1:n}) + \tilde{\eta}_{1:n}^T \Sigma_{\mathsf{RW}}^{-1} \tilde{\eta}_{1:n} \right],$$

where $\tilde{\eta}_{1:n}$ is assumed to be a column vector. Thus the penalised cost (3.4) is $\mathcal{F}_n = \min_{m,\tau_{1:m}} [\mathcal{C}(\tau_{1:m}) + m\beta]$. In the remainder of this section we will call $\mathcal{C}(\tau_{1:m})$ the cost, and $\mathcal{C}(\tau_{1:m}) + m\beta$ the penalised cost.

Whilst our cost is obtained by minimising over $\eta_{2:n}$, the following result shows that it is equal to the weighted residual sum of squares from fitting the linear model (3.7).

Proposition 4 *The cost for fitting a model with changepoints, $\tau_{1:m}$ is*

$$\mathcal{C}(\tau_{1:m}) = \min_{\Delta} (y_{1:n} - X_{\tau_{0:m}}\Delta)^T (\Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}})^{-1} (y_{1:n} - X_{\tau_{0:m}}\Delta) \tag{3.8}$$

Let $\mathcal{C}_0$ denote the cost if we fit a model with no changepoints. The following corollary, which follows from standard arguments, gives the behaviour of the cost under a null model of no changepoints. This includes a bound on the impact of mis-specifying the covariance matrix, for example due to mis-estimating the parameters of the AR(1) or random walk components of the model, or if our model for the residuals is incorrect.

Corollary 1 *Assume that data is generated from model (3.7) with $m = 0$ but with $\zeta_{1:n}$ a mean-zero Gaussian vector with $Var(\zeta_{1:n}) = \Sigma$. Let α_n^+ be the largest eigenvalue of $(\Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}})^{-1}\Sigma$. If $\Sigma = \Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}}$ then $\mathcal{C}_0 - \mathcal{C}(\tau_{1:d}) \sim \chi^2_d$. Otherwise, for any x*

$$\Pr(\mathcal{C}_0 - \mathcal{C}(\tau_{1:d}) > x) \leq \Pr(\chi^2_d > x/\alpha_n^+),$$

Furthermore, if we estimate the number of changepoints using the penalised cost (3.4) with penalty $\beta = C\alpha_n^+ \log n$ for any $C > 2$, then the estimated number of changepoints, $\hat{m}$, satisfies $\Pr(\hat{m} = 0) \to 1$ as $n \to \infty$.

To gain insight into the behaviour of the procedure in the presence of changepoints, and how it differs from standard standard change-in-mean procedures, it is helpful to consider the reduction in cost if we add a single changepoint.

Proposition 5 *Given a fixed changepoint location τ_1:*

(i) The reduction in cost for adding a single changepoint at τ_1 can be written as $\mathcal{C}_0 - \mathcal{C}(\tau_1) = (v^T y_{1:n})^2$, for some vector v defined as

$$v = \frac{1}{\sqrt{c_{\tau_1} - c_{0,\tau_1}^2/c_0}} \left\{ (\Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}})^{-1} u_{\tau_1} - \frac{c_{0,\tau_1}}{c_0} (\Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}})^{-1} u_0 \right\},$$

where u_0 is a column vector of n ones, u_{τ_1} is a column vector of τ_1 zeroes followed by $n - \tau_0$ ones, and

$$c_0 = u_0^T (\Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}})^{-1} u_0, \quad c_{0,\tau_1} = u_0^T (\Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}})^{-1} u_{\tau_1}, \quad c_{\tau_1} = u_{\tau_1}^T (\Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}})^{-1} u_{\tau_1}.$$

(ii) The vector v in (i) satisfies $\sum_{i=1}^{n} v_i = 0$ and $v^T(\Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}})v = 1$.

(iii) For any vector w that satisfies $\sum_{i=1}^{n} w_i = 0$ and $w^T(\Sigma_{\mathsf{AR}} + \Sigma_{\mathsf{RW}})w = 1$,

$$\left(\sum_{i=\tau_1+1}^{n} w_i\right)^2 \leq \left(\sum_{i=\tau_1+1}^{n} v_i\right)^2.$$

The vector v in part (i) of this proposition defines a projection of the data that is used to determine whether to add a changepoint at τ_1. The properties in part (ii) mean that this projection is invariant to shifts of the data, and that the distribution of the reduction in cost if our model is correct and there are no changes will be χ^2_1. The statistic $v^T y_{1:n}$ can be viewed as analogous to the cusum statistic (Hinkley, 1971) that is often used for a standard change-in-mean problem, and in fact if we set $\phi = 0$ and $\sigma_\eta = 0$ so as to remove the auto-regressive and random-walk aspects of the model, $|v^T y_{1:n}|$ is just the standard cusum statistic. The power of our method to detect a change at τ_1 will be governed by the distribution of this projection applied to the data in the segments immediately before and after τ_1. For a single changepoint where the mean changes by δ this distribution is a non-central chi-squared with 1 degree of freedom and non-centrality parameter $\delta^2(\sum_{i=\tau_1+1}^{n} v_i)^2$. Thus part (iii) shows that v is the best linear projection, in terms of maximising the non-centrality parameter, over all projections that are invariant to shifts in the data and that are scaled so that the null distribution is χ^2_1.

To gain insight into how the auto-regressive and random-walk parts of the model affect the information in the data about a change we have plotted different projections v for different model scenarios in the top row of Figure 3.2. The top-left plot shows the projections if we have $\phi = 0$ for different values of the random walk variance. The projection, naturally, places more weight to data near the putative changepoint, and the weight decays essentially geometrically as we move away from the putative changepoint. In the top-right plot we show the impact of increasing the autocorrelation of the AR(1) process, with the absolute value of the weight given to data points immediately before and after the putative change increasing with ϕ.

A key feature of the random walk model is that for any fixed $\sigma^2_\eta > 0$ the amount of information about a change will be bounded as we increase the segment lengths either side of the change. This is shown in the bottom-left plot of Figure 3.2 where we show the non-centrality parameter for detecting a change in the middle of the data as we vary n. For comparison we also show the non-centrality parameter of a test based on the cusum statistic (scaled so that it also has a χ^2_1 distribution under the null of no change). We can see that ignoring local fluctuations in the mean, if they exist and come from a random walk model, by using the cusum statistic leads to a reduction of power as segment lengths increase. For comparison in the bottom right we show

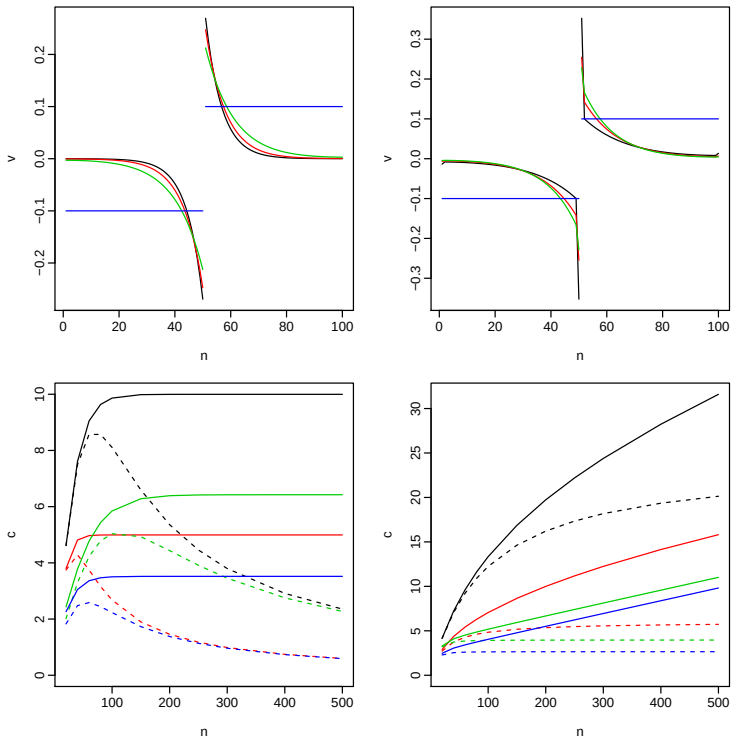


Figure 3.2: Top row: projections of data v for detecting a change in the middle of $n = 100$ data-points. Random walk model (top-left) for varying σ_η^2 of 0.03 (black), 0.02 (red) and 0.01 (green); AR(1) plus random walk model (top-right) for $\sigma_\eta^2 = 0.01$ and varying ϕ of 0.4 (black), 0.2 (red) and 0.1 (green). In both plots the blue line shows the standard cusum projection. Bottom row: non-centrality parameter for a χ_1^2 test of a change using the optimal projection (solid line) and the cusum projection (dashed line) for a change of size 1 in the middle of the data as we vary n. Out-fill asymptotics (bottom-left) where (σ_η^2, ϕ) is (0.0025,0) (black), (0.01,0) (red), (0.0025,0.5) (green) and (0.01,0.5) (blue); In-fill asymptotics (bottom-right) where for $n = 50$ (σ_η^2, ϕ) is (0.0025,0) (black), (0.01,0) (red), (0.0025,0.5) (green) and (0.01,0.5) (blue).

an equivalent comparison where we consider an infill asymptotic regime, so that as n increases we let the random walk variance decay at a rate proportion to $1/n$ and we increase the lag-1 autocorrelation appropriately. In this case using the optimal projection gives a non-centrality parameter that increases with n, whereas the cusum statistic has power that can be shown to be bounded as we increase n.

We now turn to the property of our method at detecting multiple changes. Based on the above discussion, we will consider in-fill asymptotics as $n \to \infty$.

(C1) Let $y_1, \ldots, y_n$ be generated as a finite sample from a Gaussian process on $[0, 1]$; that is $y_i = z(i/n)$ where, for $t \in [0, 1]$ $z(t) = \mu(t) + \zeta(t)$, $\mu(t)$ is a piecewise constant with m^0 changepoints at locations $r_1, \ldots, r_{m^0}$, and $\zeta(t)$ is a mean zero Gaussian process. For a given n define the true changepoint locations as $\tau_i^0 = \lfloor nr_i^0 \rfloor$. The change in mean at each changepoint is fixed and non-zero.

(C2) Assume there exists strictly positive constants c_η, c_ν and c_ϕ, such that we implement DeCAFS with $\sigma_\eta^2 = c_\eta/n$ and either (i) $\phi = 0$ and $\sigma_\nu^2 = c_\nu$; or (ii) $\phi = \exp\{-c_\phi/n\}$ and $\sigma_\nu^2 = c_\nu(1 - \exp\{-2c_\phi/n\})$.

(C3) There exists an α such that for any large enough n if Σ_n^0 is the covariance of the noise in the data generating model (C1), and $\Sigma_{\mathsf{AR}}^{(n)} + \Sigma_{\mathsf{RW}}^{(n)}$ is the covariance assumed by DeCAFS in (C2) then the largest eigenvalue of $(\Sigma_{\mathsf{AR}}^{(n)} + \Sigma_{\mathsf{RW}}^{(n)})^{-1}\Sigma_n^0$ is less than α.

The two regimes covered by condition C2 are due to the different limiting behaviour of an AR(1) model under in-fill asymptotics, depending on whether the AR(1) noise is independent, case (i), or there is autocorrelation, case (ii). The form of σ_ν^2 in each case ensures that the AR(1) process has fixed marginal variance, c_ν, for all values of n.

The key condition here is (C3) which governs how accurate the model assumed by DeCAFS is to the true data generating procedure. Clearly if the model is correct then (C3) holds with $\alpha = 1$. The following proposition gives upper bound on α in the the case where the covariance of the data generating model is that of a random walk plus AR(1) process, but with different parameter values to those assumed by DeCAFS in (C2), e.g. due to mis-estimation of these parameters.

Proposition 6 *Assume the noise process $\zeta(t)$ of the data generating process (C1) is equal to a random walk plus an AR(1) process.*

(i) If $Cov(\zeta(t), \zeta(s)) = c_\eta^0 \min(t, s)$ for $t \neq s$ and $Var(\zeta(t)) = c_\eta^0 t + c_\nu$, and DeCAFS is implemented as in (C2)(i), then (C3) holds with $\alpha = \max\{c_\nu^0/c_\nu, c_\eta^0/c_\eta\}$.

(ii) If $Cov(\zeta(t), \zeta(s)) = c_\eta^0 \min(t,s) + c_\nu^0 \exp\{-c_\phi^0 |t-s|\}$ *and DeCAFS is implemented as in (C2)(ii), then for any* $\epsilon > 0$ *(C3) holds with*

$$\alpha = \max\left\{ \frac{c_\nu^0}{c_\nu}\frac{c_\phi^0}{c_\phi}(1+\epsilon), \frac{c_\nu^0}{c_\nu}\left(1 + \frac{c_\phi}{c_\phi^0}\right)(1+\epsilon), \frac{c_\eta^0}{c_\eta} \right\}$$

The following result shows that we can consistently estimate the number of changepoints and gives a bound on the error in the estimate of changepoint locations, if we use DeCAFS under an assumption of a maximum number of changepoints (the assumption of a maximum number changes is for technical convenience, though is common in similar results, e.g. Yao, 1988).

Theorem 2 *Assume data,* $y_{1:n}$*, is generated as described in (C1), and let* $\hat{m}$ *and* $\hat{\tau}_{1:\hat{m}}$ *be the estimated number and location of the changepoints from DeCAFS implemented with parameters given by (C2), penalty* $\beta = C\alpha \log n$ *for some* $C > 2$*, and a maximum number of changes* $m_{\max} \geq m^0$*. Then as* $n \to \infty$*: if* $\phi > 0$

$$\Pr\left(\hat{m} = m^0, \max_{i=1,\ldots,m^0} \left| \hat{\tau}_i - \tau_i^0 \right| = 0 \right) \to 1;$$

and if $\phi = 0$

$$\Pr\left(\hat{m} = m^0, \max_{i=1,\ldots,m^0} \left| \hat{\tau}_i - \tau_i^0 \right| \leq (\log n)^2 \right) \to 1.$$

The most striking part of this result is the very different behaviour between $\phi = 0$ and $\phi > 0$. In the latter case, asymptotically we detect the position of the changepoints without error. This is because the positive autocorrelation in the noise across the changepoint helps us detect it. In fact, as $n \to \infty$ the signal for a change at t comes just from the lag-1 difference, $y_{t+1} - y_t$. The variance of $(y_{t+1} - y_t)$ is $O(1/n)$, and its mean is 0 except at changepoints, where it takes a fixed non-zero value. A simple rule based on detecting a change at t if and only if $(y_{t+1} - y_t)^2$ is above some threshold, $c_1(\log n)/n$ for some suitably large constant c_1, would consistently detect the changes. For the infill asymptotics we consider, empirically DeCAFS converges to such an approach as $n \to \infty$.

The theorem also gives insight into the choice of penalty β. It is natural to choose this to be the smallest value that ensures consistency, as larger values will mean loss of power for detecting changes. Assuming the DeCAFS model is correct and we have the true hyper-parameters this suggests using $\beta = 2 \log n$, the infimum of the penalties that are valid according to the theorem. This is the value that we use within our simulation study – though slightly inflating the penalty may be beneficial to account for error in the estimated hyper-parameters or if we want to account for substantial model error.

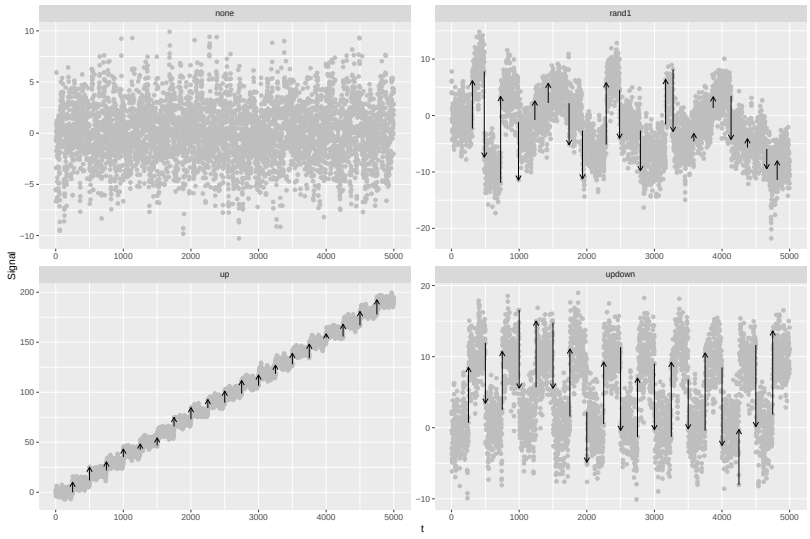


Figure 3.3: Four different change scenarios. Top-left, no change present, top-right, change pattern with 19 different changes, bottom-left up changes only, bottom-right, up-down changes of the same magnitude. In this particular example data were generated from an AR model with $\phi = 0.7,\ \sigma_\nu = 2$.

3.6 Simulation Study

3.6.1 Comparison with Changepoint Methods

We now assess the performances of our algorithm in a simulation study on four different change scenarios, illustrated in Figure 3.3. In all cases we run DeCAFS with $\beta = 2\log n$, and estimate the parameters $\phi,\ \sigma_\eta,\ \sigma_\nu$ as described in Section 3.4.

Simulations were performed over a range of evenly-spaced values of $\phi,\ \sigma_\eta,\ \sigma_\nu$. There are no current algorithms that directly model local fluctuations in the mean, so we compare with two approaches the assume a constant mean between changes: FPOP (Maidstone et al., 2017) which also assumes IID noise, and AR1Seg (Chakar et al., 2017) that models the noise as an AR(1) process. We compare default implementation of each method, which involves robust estimates of the assumed model parameters. We also compare an implementation of FPOP with an inflated penalty (Bardwell et al., 2019) to account for the autocorrelated noise. To see the impact of possible misestimation of the model parameters, we also implement DeCAFS and AR1Seg using the true parameters when this is possible.

We focus on the accuracy of these methods at detecting the changepoints. We deem a predict change as correct if it is within ± 2 observations of a true changepoint. As a measure of accuracy we use the F1 score, which is defined as the harmonic mean

of the precision (the proportion of detected changes which are correct) and the recall (the proportion of true changes that are detected). The F1 score ranges from 0 to 1, where 1 corresponds to a perfect segmentation. Separate figures for precision and recall can be found in Section A.5. Results reported are based over 100 replications of each simulation experiment, with each simulated data having $n = 5000$.

In Figure 3.4A we report performances of the various algorithms as we vary ϕ for fixed values of $\sigma_\nu = 2$ and $\sigma_\eta = 0$. In Figure 3.4B, we additionally fix $\phi = 0.85$, but we vary the size of changes. In these cases there is no random walk component and the model assumed by AR1Seg is correct.

There are a number of conclusions to draw from these results. First we see that the impact of estimating the parameters on the performance of DeCAFS and AR1Seg is small. Second, we see that using a method which ignores autocorrelation but just inflates the penalty for a change does surprisingly well unless the autocorrelation is large, $\phi > 0.5$, this is inline with results on the robustness of using a square error cost for detecting changes in mean (Lavielle and Moulines, 2000). For high values of ϕ, DeCAFS is the most accurate algorithm. The one exception are the simulations where there are no changes: the default penalty choice for AR1Seg is such that it rarely introduces a false positive.

In Figure 3.4C we explore the effect of local fluctuations in the mean by varying σ_η. We see a quick drop off in performance for all methods as σ_η increases, consistent with the fact that it is harder to detect abrupt changes when the local fluctuations of the mean are greater. Across all experiments, DeCAFS was the most accurate algorithm.

One word of caution when fitting the full DeCAFS model, is that when σ_η is large it can be difficult to estimate the parameters, as a model with a very high random walk variance produces data similar to that of a model with constant mean but high autocorrelation. Whilst the impact on detecting changes of any errors when estimating the parameters is small, it can lead to larger errors in the estimate of the signal, μ_t: as different parameter estimates mean that the fluctuations in the data are viewed as either fluctuations in the noise process or in the signal. An example of this is shown in Section A.4.1.

3.6.2 Robustness to Model Mis-specification

We now investigate the performance of DeCAFS when its model is incorrect. First we follow Chakar et al. (2017) and simulate data with a constant mean between changes but with the noise process being AR(2), i.e. $\epsilon_t = \phi_1\epsilon_{t-1} + \phi_2\epsilon_{t-2} + \nu_t$. In Figure 3.5 we report F1 Scores for DeCAFS and AR1Seg as we vary range ϕ_2. Obviously as $|\phi_2|$ increases, all algorithms perform worse, but the segmentations returned from DeCAFS are the more reliable as we increase the level of model error.

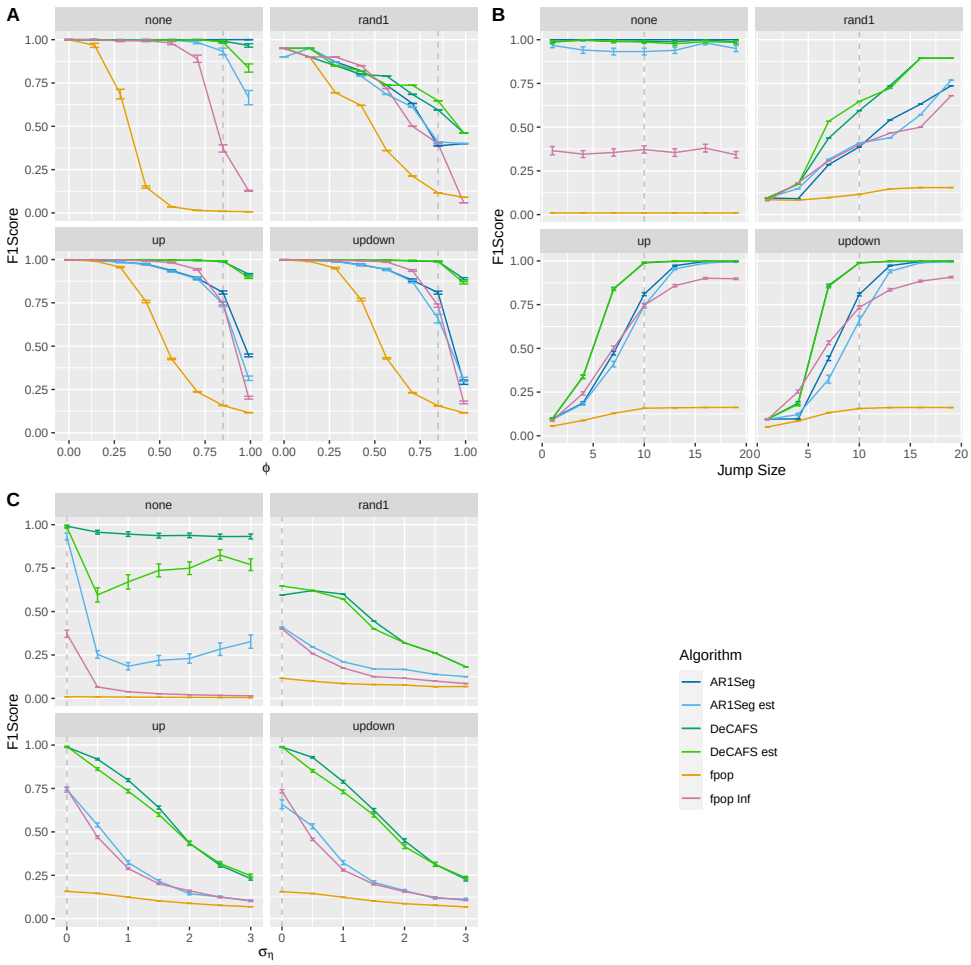


Figure 3.4: F1 Scores on the 4 different scenarios. In **A** a pure AR(1) over a range of values of ϕ, for fixed values of $\sigma_\nu = 2$, $\sigma_\eta = 0$ and a change of magnitude 10. In **B** a pure AR(1) process with fixed $\phi = 0.85$ and changes in the signal of various magnitudes. In **C** the full model with $\phi = 0.85$ for a range of values of σ_η. The grey line represent the cross-section between parameters values in **A**, **B** and **C**. AR1Seg est. and DeCAFS est. refer to the segmentation of the relative algorithms with estimated parameters. Note, in **B** the results from DeCAFS and DeCAFS est overlap so only one line is visible. Other algorithms use the true parameter values.

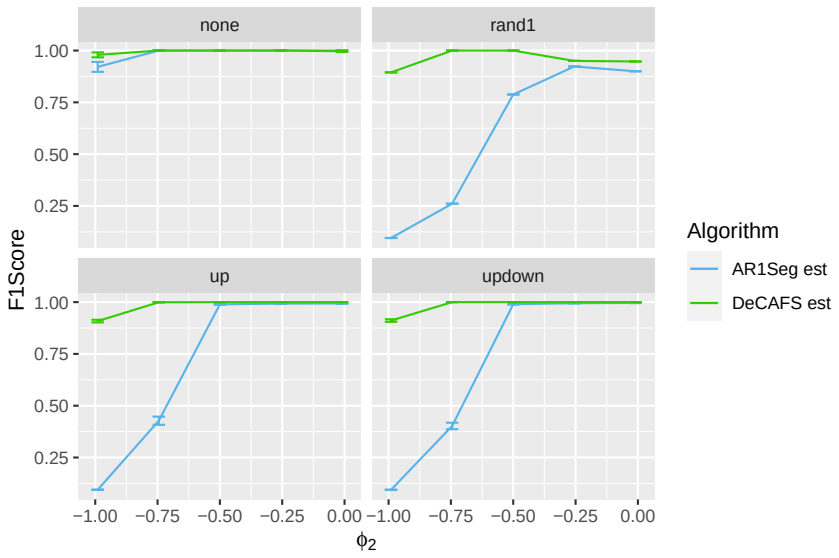


Figure 3.5: F1 score on different scenarios with AR(2) noise as we vary ϕ_2. Data simulated fixing $\sigma_\nu = 2$, $\sigma_\eta = 0$ and $\phi_1 = 0.3$ over a change of size 20.

Second, we consider local fluctuations in the mean that are generated by a sinusoidal process rather than the random walk model, see Figure 3.6B. In Figure 3.6A we compare performance of DeCAFS and AR1Seg as we vary the frequency of the sinusoidal process. Again we see that DeCAFS gives more reliable segmentations in these cases. In the three change scenarios performance decrease as we increase the frequency of the process. In these cases it becomes significantly harder to detect any changepoints, however DeCAFS still has higher scores than AR1Seg since it is more robust and returns fewer false positives. Additional simulation results, showing the robustness of DeCAFS to the mean fluctuations being from an Ornstein-Uhlenbeck process or to the noise being AR(1) within a segment but independent across segments are shown in Sections F and G in the Supplementary Material.

3.6.3 Comparison to LAVA

The LAVA method of Chernozhukov et al. (2017) can be applied to model a signal as the sum of a piecewise constant function and a locally fluctuating function: and is thus a natural alternative to DeCAFS. If we let X denote the $n \times n$ matrix whose ith column has $i-1$ zeroes followed by $n-i+1$ ones then LAVA can estimate the mean $\mu_{1:n}$ as $X(f_{1:n} + g_{1:n})$ where the vectors f and g minimise

$$(y - X(f_{1:n} + g_{1:n}))^T (y - X(f_{1:n} + g_{1:n})) + \lambda_1 ||f_{1:n}||_1 + \lambda_2 ||g_{1:n}||_2^2,$$

with $||\cdot||_1$ and $||\cdot||_2$ denoting, respectively, the L_1 and L_2 norms.

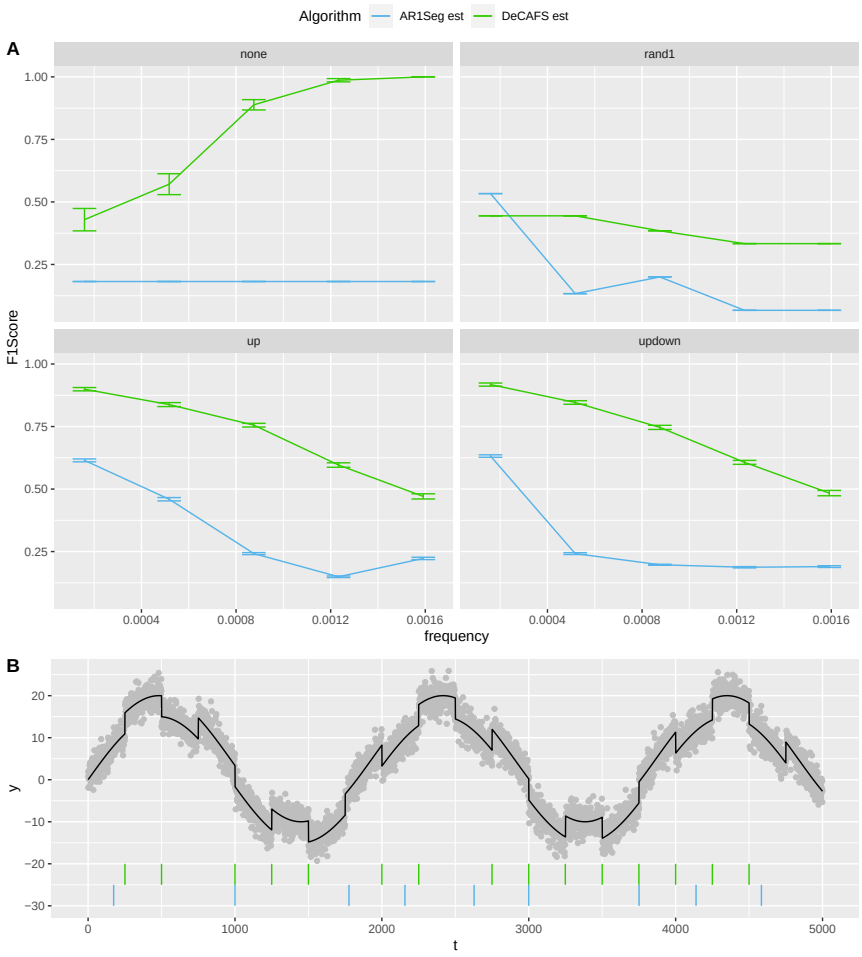


Figure 3.6: In **A** the F1Score on the 4 scenarios for the Sinusoidal Model for fixed amplitude of 15, changes of size 5 and IID Gaussian noise with a variance of 4, as we vary the frequency of the sinusoidal process. In **B** an example of a realization for the updown scenario, vertical segments refer to estimated changepoint locations of DeCAFS (in light green) and AR1Seg (in blue).

The interpretation of this is that $(f + g)_i$ is the change in the mean of the data from time $i - 1$ to time i. The penalties are such that $f_{1:n}$ is sparse, and thus is modelling the abrupt changes in the mean, while $g_{1:n}$ is dense and is accounting for the local fluctuations. It is possible to show that DeCAFS is equivalent to LAVA if we have independent noise and replace the L_1 norm by an L_0 norm.

We implemented LAVA using the `lavash` package in `R`. This implementation of LAVA is substantially slower than DeCAFS, and empirically the computational cost appears to increase with the cube of the number of data points. As a result we compare DeCAFS with LAVA on simulated data of length $n = 1000$ (for which LAVA takes, on average, 132 seconds and DeCAFS 0.02 seconds per dataset).

LAVA tunes λ_1 by cross-validation. We used a plug-in value for λ_2 based on the oracle choice suggested in Chernozhukov et al. (2017). This choice depends on the variance of the local fluctuations, i.e. the variance of the random walk component in our model, and we implemented LAVA using both the true variance (denoted LAVA), and using the same estimate of the variance as we use for DeCAFS (denoted LAVA_est). The plug-in approach seemed to give better results, and was substantially faster than using cross-validation to tune λ_2. It also makes a comparison with DeCAFS easier, as in situations where neither method estimates any abrupt changes, the two methods then give essentially identical estimates of the mean.

Results from analysing data simulated under a pure random-walk model (so $\phi = 0$) are shown in Figure 3.7. Whilst both method perform similarly at estimating the underlying mean, we see that LAVA is less reliable at estimating the changepoint locations – and often substantially over-estimates the number of changepoints. As summarised in the introduction, this is not unexpected as it is common for methods that use ℓ_1 penalties on the size of an abrupt change to overestimate the number of changes (see e.g. Fearnhead et al., 2018; Jewell et al., 2020).

3.7 Gene Expression in *Bacilus subtilis*

We now evaluate DeCAFS on estimating the expression of cells in the bacteria *Bacilus subtilis*. Specifically we analyze data from Nicolas et al. (2009), which is data from tiling arrays with a resolution of less than 25 base pairs. Each array contains several hundred thousand probes which are ordered according to their position on the bacterial chromosome. For a probe, labelled t say, we get an RNA expression measure, Y_t. Figure 3.8 shows data from 2000 probes. Code and data used in our analyses, presented below, are available on forgemia : `https://forgemia.inra.fr/guillem.rigaill/decafsrna`.

The underlying expression level is believed to undergo two types of transitions, large changes which Nicolas et al. (2009) call shifts and small changes which they call drifts. Thus it naturally fits our modelling framework of abrupt changes, the

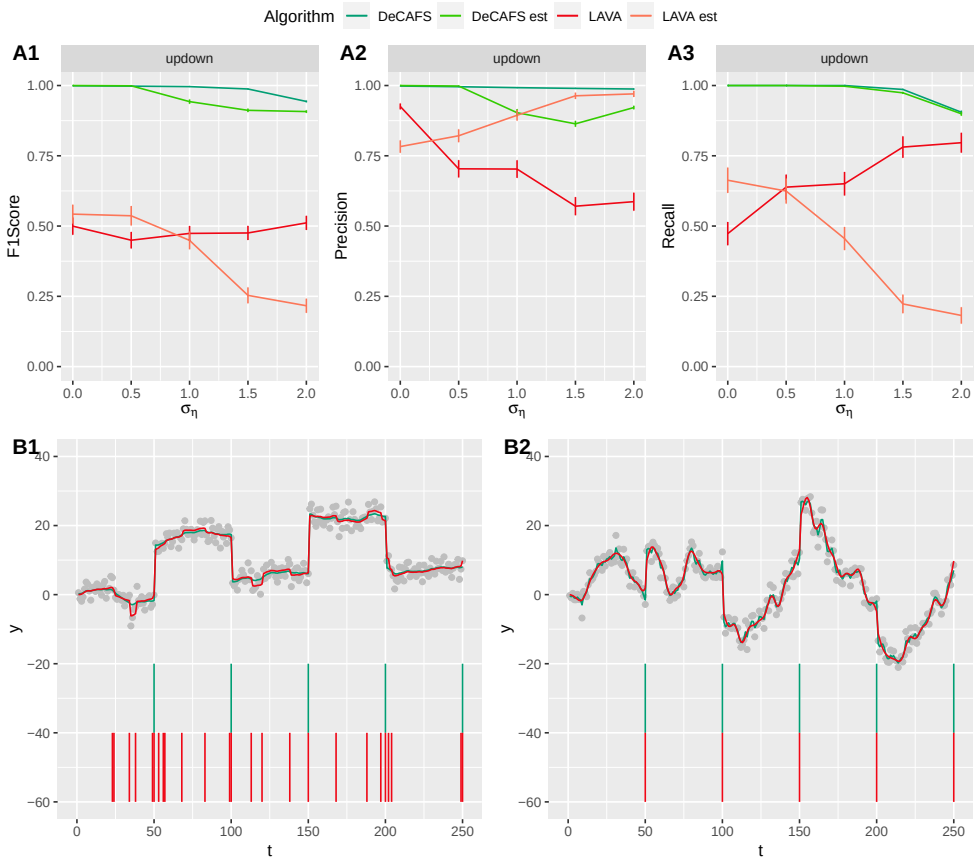


Figure 3.7: On top: comparison of the F1 Score in **A1**, Precision in **A2** and MSE in **A3**, for DeCAFS (in green) and LAVA (in red) with oracle initial parameters and the relative results with estimated initial parameters (in lighter colours), on the updown scenario for a random walk signal over a range of values of σ_η. On the bottom the first 250 observations of two realization of the experiment with, in **B1**, σ_η equal to 0.5 and in **B2** σ_η equal to 2. Again, the continuous line over the data points represent the relative signal estimations of DeCAFS and LAVA; the segments their changepoint locations estimates.

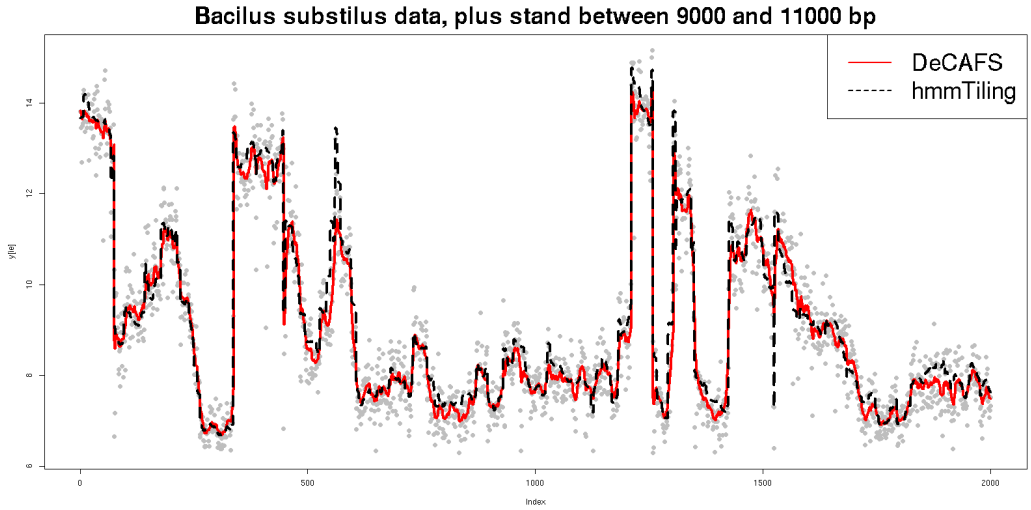


Figure 3.8: Data on 2000 bp of the plus-strand of the *Bacilus subtilis* chromosome. Grey dots show the original data. The plain red line represents the estimated signal of DeCAFS with a penalty of $10\log(n)$. The dashed black line represents the estimated signal of hmmTiling.

shifts, between which there are local fluctuations caused by the drifts. To evaluate the performance of DeCAFS at estimating how the gene expression levels vary across the genome we will compare to the hmmTiling method of Nicolas et al. (2009). This method fits a discrete state hidden Markov model to the data, with the states being the gene expression level, and the dynamics of the hidden Markov model corresponding to either drifts or shifts. As a comparison of computational cost for of the two methods, DeCAFS takes about 7 minutes to analyse data from one of the strands, each of which contains around 192,000 data points. Nicolas et al. (2009) reported a runtime of 5 hours and 36 minutes to analyse both strands.

A comparison of the estimated gene expression level from DeCAFS and from hmmTiling, for a 2000 base pair region of the genome, is shown in Figure 3.8. We see a close agreement in the estimated level for most of the region, except for a couple of regions where hmmTiling estimates abrupt changes in gene expression level that DeCAFS does not.

To evaluate which of DeCAFS and hmmTiling is more accurate, we follow Nicolas et al. (2009) and see how well the estimated gene expression levels align with bioinformatically predicted promoters and terminators. A promoter roughly corresponds to the start of a gene, and a terminator the end, and we expect gene expression to increase around a promoter and decrease around a terminator.

For promoters, consider all probe locations t from the tiling chip and consider a

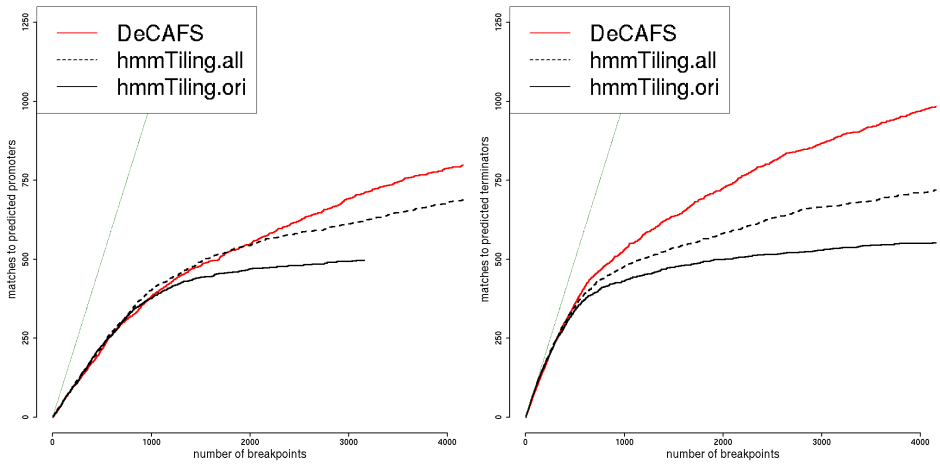


Figure 3.9: Benchmark comparisons. The number of promoters (left) and terminators (right) correctly predicted on the plus strand, $M(\delta)$ using a 22 bp distance cutoff, as a function of the number of predicted breakpoints, $R(\delta)$. Plain black lines are the results of hmmTiling (as reported in Figure 4 of Nicolas et al., 2009)). Dotted black lines are the results of hmmTiling when considering all probes rather than only those called transitions. Plain red lines are the results of DeCAFS using $\beta = 8\log(n)$ for promoters and $5\log(n)$ for terminators. These values were learned on the minus strand using a data-driven approach.
The thin dark-green leaning line represent $y = x$.

threshold parameter δ. We can count the number of probe locations with a predicted difference $\hat{d}_t = \hat{\mu}_{t+1} - \hat{\mu}_t$ strictly greater than δ. We call this $R(\delta)$. Among those probes, we can count how many have a promoter nearby (within 22 base pairs). We call this $M(\delta)$. By symmetry we can define an equivalent measure for terminators. A method is better than another if for the same $R(\delta)$ it achieves a larger $M(\delta)$.

We used a data-driven approach to choose the penalty, β for DeCAFS, benefitting from having separate data from the plus and minus strand of the chromosome. For Figure 3.9 the penalty was learned on the minus strand data and tested on the plus strand data. More specifically we ran DeCAFS on the minus strand for $\beta = \{2\log(n), 2.5\log(n)...30\log(n)\}$. For each β we computed $M(\delta)$ for a fixed $R(\delta) = 750$ and took the β maximizing $M(\delta)$: $8\log(n)$ for promoters and $5\log(n)$ for terminators.

Figure 3.9 plots $M(\delta)$ against $R(\delta)$ for the plus strand as we vary δ for DeCAFS and two different estimates from hmmTiling. The first, hmmTiling.ori, are the prediction presented in Nicolas et al. (2009). The second, hmmtTiling.all, are those obtained when using all probes rather than only those called transitions by hmmTiling.

In the case of promoters the prediction of hmmTiling is slightly better than

DeCAFS for lower thresholds but noticeably worse for higher thresholds. In the case of terminators the prediction of DeCAFS are clearly better than those of hmmTiling. Given that DeCAFS was not developed to analyze such data we believe that its relatively good performances for promoters and better performances for terminators is a sign of its versatility.

Chapter 4

Functional Pruning Online Changepoint Detection

4.1 Introduction

Over the previous decade we have witnessed a renaissance of changepoint algorithms, and they can now be seen to make a difference to many real-world applications. Most of the current literature focuses on an analysis *a posteriori* of observing a series of data; such a type of analysis is often referred to as *offline* changepoint detection. However, as technology develops, the demand from several fields for *online* changepoint detection procedures has increased drastically over recent years. Examples include, but are not limited to, IT and cyber security (Jeske et al., 2018; Tartakovsky et al., 2012; Peng et al., 2004); detecting gamma ray bursts in astronomy (Fridman, 2010; Fuschino et al., 2019); detecting eathquake tremors (Popescu and Aiordăchioaie, 2017; Xie et al., 2019); industrial processes monitoring (Pouliezos and Stavrakakis, 2013); detecting adverse health events (Clifford et al., 2015); and monitoring the structural integrity of aeroplanes (Alvarez-Montoya et al., 2020; Basseville et al., 2007).

The online setting raises computational challenges that are not present in offline changepoint detection. A procedure needs to be sequential, in the sense that one should process the observations as they become available, and at each iteration one should make a decision whether to flag a changepoint based on the information to date. The procedure needs to be able to run on a finite state machine for an indefinite amount of iterations, *i.e.* be constant in memory; and it needs to be able to process observations, at least on average, as quickly as they arrive. Many online changepoint application settings have high frequency observations, and some also have limited computational resources. For example, the observations from ECG data in the 2015 PhysioNet challenge (Clifford et al., 2015) are sampled at 240Hz, while methods for

detecting gamma ray bursts (Fuschino et al., 2019) need to process high-frequency observations and be able to be run on small computers on board micro-satellites. A challenge with online changepoint algorithms is to meet this computational constraints whilst still having close to optimal statistical properties.

This paper considers the univariate change in mean problem. Current online changepoint methods with a linear computational cost include the method of Page (1955) that assumes we know both the pre-change and post-change mean; or moving window methods such as MOSUM. Whilst assuming the pre-change mean is known is reasonable in many applications as there will be substantial data to estimate this mean (though see discussion in Gösmann et al., 2019), the former approach can lose power if the assumed size of change is wrong. For example, this method can have almost no power to detect changes that are less than half the size of the assumed change. Similarly, moving window methods can perform poorly if the window size is inappropriate for the size the change. A small window size will mean little power at detecting small changes, whilst too large a window will lead to delays in detecting larger changes. See Section 4.2.1 for an example of these issues.

An alternative, with more robust statistical properties, is to e.g. apply a moving window approach but consider all possible window sizes. In the known pre-change mean setting this is known as the Page-cusum approach (Kirch et al., 2018) and is the approach of Yu et al. (2020) for the case of an unknown pre-change mean. The theoretical results in Yu et al. (2020) demonstrate the excellent statistical properties of such a method. However current exact implementations of this idea have a computational cost per iteration that is linear in the number of observations, and thus have an overall quadratic computational cost. Yu et al. (2020) comment on the challenge of developing faster algorithm with good statistical guarantees: "we are not aware of nor expect to see any theoretically-justified methods with linear order computational costs". This paper presents such an algorithm, which we call Functional Online CuSUM (FOCuS). We develop FOCuS for detecting changes in mean in univariate data, and it can be applied to settings where either the pre-change mean is known or unknown. In both cases it has an average per iteration computational cost that increases with the logarithm of the number of observations. Furthermore, we develop an approximate version of FOCuS which empirically has almost identical performance and has bounded cost per iteration. In the unknown pre-change mean case FOCuS implements the method of Yu et al. (2020), and our implementation of FOCuS can analyse 1 million observations in less than a second on a common personal computer.

Much research on online changepoint methods has looked at how to implement methods so that they have well characterised performance under the null hypothesis of no change. There are two distinct criteria for quantifying a method's behaviour under the null, one is the average run length (Reynolds, 1975) which is the expected

number of observations until we detect a change. The other is a significance level – the probability of ever detecting a change if the method is run on infinitely long data. In practice these two criteria effect the choice of threshold for a detection method. If we wish to control the significance level then we need a threshold that increases with the number of observations (see e.g. Kirch and Kamgaing, 2015), whereas if we wish to control the average run length we can use a fixed threshold. The FOCuS algorithm can be used with either approach – but for simplicity we will only use a fixed threshold in this paper.

The outline of the paper is as follows. In Section 4.2 we consider the challenge of detecting a univariate change in mean when the pre-change mean is assumed known. We present the FOCuS algorithm which can be viewed as implementing the procedure of Page (1955) simultaneously for all possible size of change. Our main theoretical result shows that FOCuS achieves this with an average computational cost per iteration that is logarithmic in the number of data points. Our bound on the average per iteration cost is tight, and when processing the one millionth observation roughly equates to the cost of evaluating 15 quadratics. In Section 4.3 we then give two extensions of the FOCuS algorithm. First to the case where the pre-change mean is unknown. This algorithm can be viewed as implementing the statistical tests of Yu et al. (2020): but whereas their algorithms are either exact but with a linear cost per iteration or approximate with a cost that is logarithmic in the number of iterations, the FOCuS algorithm is exact and has an average cost that is logarithmic per iteration. We do not present any statistical theory for FOCuS , but this is covered in Yu et al. (2020). Second we show how to extend FOCuS to detecting changes in the presence of outliers. In Section 4.4 we show a monitoring application for FOCuS on some AWS Clowdwatch server instances. Finally, the paper concludes with a discussion.

4.2 Known pre-change mean

4.2.1 Problem Set-up and Background

Consider the problem of detecting a change in mean in univariate data. We will let x_t denote the data at time t, for $t = 1, 2, \ldots$. We are interested in online detection, that is after observing each new data point we wish to decide whether or not to flag that a change has occurred. We will first assume that the pre-change mean is known. Often the methods below are implemented in practice using a plug-in estimator for the pre-change mean that is calculated from training data.

A common approach to this problem (see Kirch et al., 2018) is to use a cumulative sum of score statistics, also know as a CUSUM based procedure. Assume we model our data as coming from a parametric model with density $f(x; \mu)$ and denote the

pre-change mean as μ_0. Define the score statistic of an observation x as

$$H(x, \mu) = \frac{\partial \log f(x; \mu)}{\partial \mu}.$$

Then if there is no change prior to time n

$$\mathrm{E}\left(H(X_i, \mu_0)\right) = 0, \text{ for } i = 1, \ldots, n.$$

Thus evidence of a change prior to time n can be obtained by monitoring the absolute values of partial sums of these score statistics, which we denote as

$$S(s, n) = \sum_{i=s+1}^{n} H(x_i, \mu_0).$$

The idea is that these partial sums should be close to 0 if there is no change, and they should diverge away from zero if there is a change.

For ease of presentation, and to make ideas concrete, in the following we will consider the case where we have a Gaussian model for the data. In this case $H(x, \mu) = (x - \mu)$. Also as we are assuming μ_0 is known, then without loss of generality we can set $\mu_0 = 0$.

There have been a number of different choices of partial sums that we can monitor. For detecting a change after observing x_n, Kirch et al. (2018) mention the following statistics

$$\text{CUSUM} \qquad C(n) = \frac{1}{\sqrt{n}}|S(0, n)|; \tag{4.1}$$

$$\text{MOSUM} \qquad M_w(n) = \frac{1}{\sqrt{w}}|S(n - w, n)|; \tag{4.2}$$

$$\text{mMOSUM} \qquad mM_k(n) = \frac{1}{\sqrt{nk}}|S(n - \lfloor kn \rfloor, n)|; \tag{4.3}$$

$$\text{Page-CUSUM} \qquad P(n) = \max_{0 \leq w < n} \frac{1}{\sqrt{w}}|S(n - w, n)|. \tag{4.4}$$

The scale factor in each case is to normalise the cumulative sum $S(\cdot, \cdot)$ so as to standardise its variance. In each case we would compare the statistic at time n with some appropriate threshold, and detect a change prior to n if the statistic is above the threshold. As discussed in the introduction the choice of threshold impacts the properties of the test under the null. It is possible to choose thresholds that are constant or that increase as n increases (Kirch et al., 2018), but for simplicity we will use constant thresholds throughout.

The standard CUSUM statistic uses the partial sum of score statistics to time n. For both the MOSUM procedure (Eiauer and Hackl (1978), Chu et al. (1995)) and

mMOSUM procedure (originating in Chen and Tian, 2010) we need to specify a tuning parameter. The MOSUM method uses the partial sum over a window of the most recent $w > 0$ observations, whilst the mMOSUM fixes some proportion $0 < k < 1$ and uses the partial sum over the most recent proportion k of the observations. All three of these statistics are online, in that there is again only an $O(1)$ update of the statistics as we process each new data point. The Page-CUSUM maximises over all possible partial sums ending at time n.

To understand the difference between CUSUM, MOSUM and Page-CUSUM we implemented these methods for detecting a change at time t to some mean μ_1 for different values of t and μ_1 and compared the detection delay of each statistics. Results are shown in Figure 4.1. In Figure 4.1a, we simulated data with a change after 1000 observations and the size of change is chosen to give high power for the window size of the MOSUM procedure. MOSUM and Page-CUSUM tend to detect a change quickly. In the second example, shown in Figure 4.1b, we reduce the magnitude of the change, and find that the MOSUM test loses power substantially and has a much larger detection delay than Page-CUSUM. In our final example, see Figure 4.1c, we have the same size of change as the first example, but now the change occurs after 8,000 observations. In this case we see that the CUSUM statistic behaves poorly. This is because the CUSUM statistic has to average the signal from data after the change with all the data prior to change, and this reduces the power of the test statistic, particularly when there is substantial data pre-change. Both MOSUM and Page-CUSUM perform as in the first example.

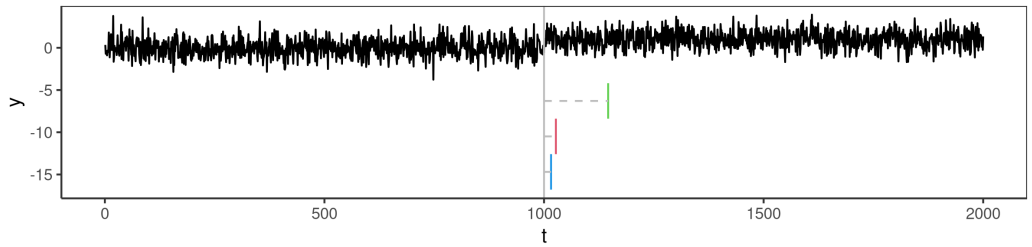
Whilst we do not show the performance of mMOSUM in these examples, it shares a similar sensitivity to choice of window proportion, k, as the MOSUM does for window size.

The Page-CUSUM tries to avoid the issues with choosing a window size within the MOSUM method, and is equivalent to maximising the MOSUM statistic over w. However current implementations of Page-CUSUM are not online – in fact the computational cost of calculating $\max_{0 \leq s < n} |S(s, n)|$ increases linearly with n, resulting in an $\mathcal{O}(n^2)$ computational complexity.

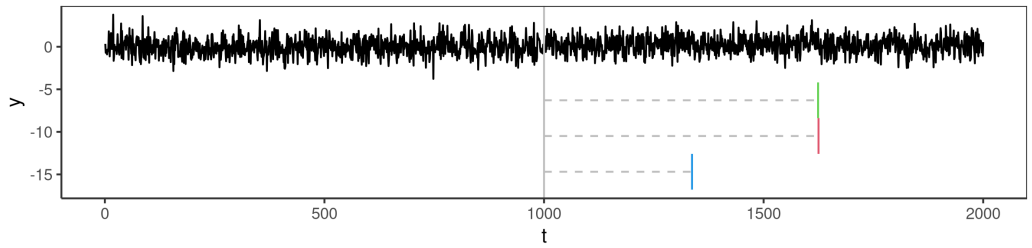
An alternative approach (Page, 1954, 1955) to detecting the change is based on sequentially applying a likelihood ratio test under an assumed value for the post-change mean, μ_1. Under our Gaussian model with pre-change being 0, we have that the contribution to the likelihood-ratio statistic from a single data point, x_t is

$$LR(x_t, \mu_1) = \mu_1 \left(\frac{\mu_1}{2} - x_t \right) \tag{4.5}$$

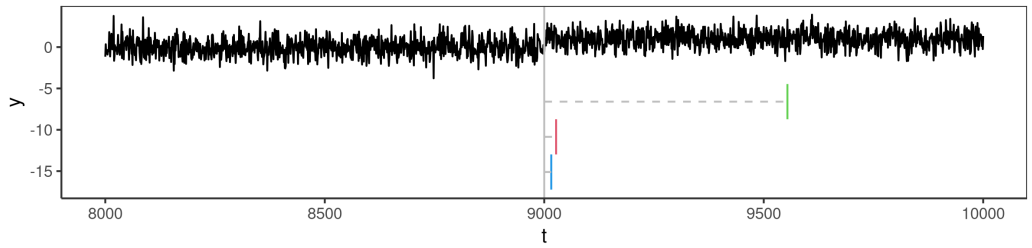
At time n, the test-statistic for a change at time s is the sum of these terms from



(a)



(b)



(c)

Figure 4.1: Detection delays of CUSUM (in green), MOSUM (in red, with $w = 50$) and Page-CUSUM (in blue) on three sequences. The sequences were generated in the following way: (a) a sequence of 2000 observations with a change of size 1 at 1000; (b) similar to (a) but with a change in the mean of 0.2; (c) similar (a) again, but with an additional 8×10^3 observations at the start of the sequence. Penalties were tuned accordingly to the simulation study in Section 4.2.3. The solid grey line refers to the true changepoint location, the dashed segments to the detection delays of the super-mentioned procedures.

$t = s+1, \ldots, n$. As we do not know the time of the change, we maximise over s:

$$\mathcal{Q}_{n,\mu_1} = \max_{0 \leq s < n} \sum_{t=s+1}^{n} \mu_1 \left(\frac{\mu_1}{2} - x_t \right).$$

We will call this statistic the *sequential-Page statistic*.

Whilst our definition of $\mathcal{Q}_{n,\mu_1}$ involves a sum over n terms, Page (1954) showed that we can calculate $\mathcal{Q}_{n,\mu_1}$ recursively in constant time as:

$$\mathcal{Q}_{n,\mu_1} = \max \left\{ 0,\ \mathcal{Q}_{n-1,\mu_1} + \mu_1 \left(\frac{\mu_1}{2} - x_t \right) \right\}. \tag{4.6}$$

One issue with the sequential-Page statistic is the need to specify μ_1, and a poor choice of μ_1 can substantially reduce the power to detect a change. This is similar to the choice of window size for MOSUM. To partially overcome this, in both cases we can implement the methods multiple times, for a grid of either window sizes or values of μ_1. Obviously, this comes with an increased computational cost.

4.2.2 FOCuS^0 : solving the Page recursion for all μ_1

Our idea is to solve the sequential-Page recursion simultaneously for all values of the post-change mean. That is we can re-write (4.6) in terms of a recursion for a function $Q_n(\mu)$ of the post-change mean $\mu_1 = \mu$. We then have $Q_0(\mu) = 0$ and for $n = 1, \ldots,$

$$Q_n(\mu) = \max \left\{ 0,\ Q_{n-1}(\mu) + \mu \left(x_n - \frac{\mu}{2} \right) \right\}. \tag{4.7}$$

We would then use $\max Q_n(\mu)$ as our test statistic. It is straightforward to see that for any μ_1, $Q_n(\mu_1) = \mathcal{Q}_{n,\mu_1}$. Thus if we can efficiently calculate $Q_n(\mu)$ then our test statistic is equivalent to the maximum value of the sequential-Page statistic over all possible choices of post-change mean.

Furthermore, the following proposition shows that this test statistic is equivalent to the Page-CUSUM statistic (4.4), or equivalently the maximum of the MOSUM statistic (4.2) over all possible windows.

Proposition 7 *The maximum of $Q_n(\mu)$ satisfies*

$$\max_{\mu} Q_n(\mu) = \frac{1}{2} P(n)^2 = \frac{1}{2} \max_{w} M_w(n)^2,$$

where $P(n)$ is the Page-CUSUM statistic and $M_w(n)$ is the MOSUM statistic with window size w.

Algorithm 7: FOCuS^0 (one iteration)

Data: x_n the data at time n; $Q_{n-1}(\mu)$ the cost function from the previous iteration.

Input: $\lambda > 0$

$Q_n(\mu) \leftarrow \max\left\{0,\ Q_{n-1}(\mu) + \mu\left(x_n - \frac{\mu}{2}\right)\right\}$; `// Algorithm 8 : amortized` $O(1)$

$\mathcal{Q}_n \leftarrow \max_\mu Q_n(\mu)$; `// Theorem 5 : average` $O(\log(n))$

if $\mathcal{Q}_n \geq \lambda$ **then**

  return n *as a stopping point*;

end

return $Q_n(\mu)$ *for the next iteration.*

The proof for this can be found in Appendix B.1.

A skeleton of the resulting algorithm for online changepoint detection is given in Algorithm 7. We call this the Functional Online CuSUM (FOCuS) algorithm. To be able to distinguish this version, that assumes a known pre-change mean, we call Algorithm 7 FOCuS^0 . The FOCuS^0 algorithm is only useful if it is computationally efficient, and in particular if we can implement Steps 1 and 2 efficiently. These steps correspond to solving recursion (4.7) to get $Q_n(\mu)$ from $Q_{n-1}(\mu)$ and then maximising $Q_n(\mu)$. We will describe each of these steps in turn, and present results on their average computational cost.

4.2.2.1 Step 1: updating the intervals and quadratics

For Step 1 of Algorithm 7 we propose to update the value of $Q_n(\mu)$ separately for $\mu > 0$ and $\mu < 0$. These can be updated in an identical manner, so we will only describe the update for $\mu > 0$. We will use the fact that (4.7) maps piecewise quadratics to piecewise quadratics, and hence $Q_n(\mu)$ will be piecewise quadratic (see Maidstone et al., 2017, for a similar idea) and can be stored as a list of ordered intervals of μ together with the co-efficients of the quadratic for $Q_n(\mu)$ on that interval. Let $S_t = \sum_{j=1}^t x_j$ be the sum of the first t data points. The quadratic introduced at iteration τ will be of the form

$$\mu\left(\sum_{t=\tau+1}^{n} x_t - (n-\tau)\frac{\mu}{2}\right) = \mu\left((S_n - S_\tau) - (n-\tau)\frac{\mu}{2}\right).$$

Thus, if at time n we know n and S_n, its co-efficients can be calculated if we store τ and S_τ. This information stored for the quadratic does not need to be updated at each iteration.

As a result, to update $Q_n(\mu)$ we need only add a quadratic, update the interval associated with currently stored quadratics, and remove quadratics that are not longer optimal for any μ (i.e. whose associated interval is the empty set). Intervals only change due to comparisons between stored quadratics and the new quadratic. The new quadratic is a constant at 0. A key observation, simplifying the functional update, is that the difference between a quadratic introduced at iteration τ and 0 gives that the new quadratic is better on the interval

$$\left[2\sum_{t=\tau+1}^{n}\frac{x_t}{(n-\tau)}\ ,\ +\infty\right). \tag{4.8}$$

Considering all τ we get get that the new quadratic is better than all others on

$$\left[2\max_{\tau}\sum_{t=\tau+1}^{n}\frac{x_t}{(n-\tau)}\ ,\ +\infty\right).$$

Therefore to update $Q_n(\mu)$ we essentially need to recover

$$\max_{\tau}\sum_{t=\tau+1}^{n}\frac{x_t}{(n-\tau)}. \tag{4.9}$$

Furthermore this argument shows that the lower bound for any existing quadratics is either unchanged, if it is lower than this value, or that quadratic can be removed.

One can show that the set of changes in $Q_n(\mu)$ are part of the convex hull of the 2D points $\{(1, S_1), (2, S_2), \cdots, (n, S_n)\}$, see Lemma 3 in the Appendix. In fact, our algorithm for updating the list of quadratics is based on Melkman's Algorithm (Melkman, 1987) for calculating the the convex hull of a set of points, and is given in Algorithm 8. A graphical representation of 4 iterations of the procedure is found in Figure 4.2. As described above, $Q_n(\mu)$ is defined by k quadratics, with associated triples denotes as (τ_i, s_i, l_i) for $i = 1, \ldots, k$. These are ordered so that $0 = l_1 < \cdots < l_k$. The idea is that at time $n + 1$ we need to find the value of l such that $Q_n(l) = 0$, by solving (4.9). We can do this by considering each quadratic in turn, starting with kth quadratic and stepping through them in decreasing order. If we are considering the ith quadratic we check whether $Q_n(l_i) < 0$ or not. If it is we remove the quadratic and move to the $(i-1)$th quadratic (or stop if $i = 1$). If $Q_n(l_i) > 0$ then $l > l_i$ and we find l as the positive value of μ such that the ith quadratic is equal to 0.

We can show that Algorithm 8 has an amortized per-iteration cost that is $O(1)$. The intuition is that each quadratic is added once and removed once, and otherwise unchanged. Thus the average per-iteration cost is essentially the cost of adding and of removing a quadratic.

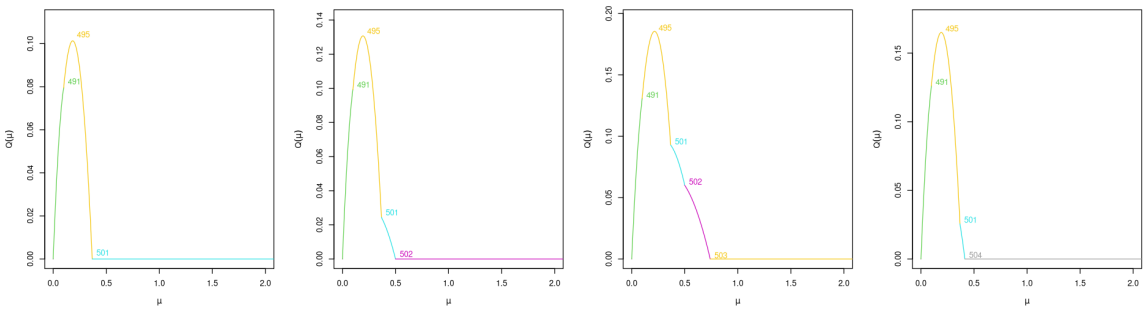


Figure 4.2: A graphical representation of FOCuS cost function across 4 iterations from time 500 to 504 (left to right). Labels report the iterations where the quadratics were added. The line being the newly added quadratic (hence yet to be updated). We notice how the quadratic introduced at time 502, in purple, is pruned after just two iterations, being no longer optimal.

Algorithm 8: Algorithm for $\max\{0,\ Q_{n-1}(\mu) + \mu(x_n - \mu/2\}$ for $\mu > 0$

Data: $Q_n^+(\mu) = \mathrm{Q}$ an ordered set of triples $\{q_i = (\tau_i,\ s_i,\ l_i)\ \forall\ i = 1, \ldots,\ k\}$, x_n and S_{n-1}

1 $S_n \leftarrow S_{n-1} + x_n$; `// update cumulative sum`
2 $q_{k+1} \leftarrow (\tau_{k+1} = n,\ s_{k+1} = S_n,\ l_{k+1} = \infty)$; `// new quadratic`
3 $i \leftarrow k$;
4 **while** $2(s_{k+1} - s_i) - (\tau_{k+1} - \tau_i)l_i \leq 0$ *and* $i \geq 1$ **do**
5 $\quad i \leftarrow i - 1$;
6 **end**
7 $l_{k+1} \leftarrow 2(s_{k+1} - s_i)/(\tau_{k+1} - \tau_i)$; `// update new quadratic`
8 **if** $i \neq k$ **then**
9 $\quad \mathrm{Q} \leftarrow \mathrm{Q} \setminus \{q_{i+1},\ \ldots, q_k\}$; `// pruning old quadratic`
10 **end**
11 **return** $\{\mathrm{Q}, q_{k+1}\}, \quad S_n$

Theorem 3 *The worst case complexity of Algorithm 8 for any data $x_1, \ldots, x_T$ is $O(T)$ and its amortized complexity per iteration is $O(1)$.*

Proof: At iteration n, let k_n be the number of quadratics input, and let c_n be the number of times the algorithm repeats the while loop in Steps 4 to 6. Let C_1 be the cost of steps 1 to 3 and 7 to 11, and C_2 be the cost of one set of steps 4 to 6. Then the computational cost of one iteration of Algorithm 8 is $C_1 + c_n \times C_2$.

The key observation is that $k_{n+1} = k_n - (c_n - 1) + 1$. That is if we repeat Steps 4 to 6 c_n times then we will remove $c_n - 1$ quadratic in Step 9 and add one quadratic in

Step 11. Furthermore $k_1 = 0$ and k_{T+1} is the number of quadratics for $Q_T(\mu)$. Thus the total computational cost is

$$\sum_{n=1}^{T}(C_1 + c_n C_2) = C_1 T + C_2 \sum_{n=1}^{T} c_n = C_1 T + C_2 \sum_{n=1}^{T}(2 + k_n - k_{n+1}).$$

Due to the cancellations in the telescoping sum and the fact that $k_1 = 0$ we have

$$\sum_{n=1}^{T} c_n = 2T - k_{T+1} \leq 2T$$

Thus the theorem holds □

As the overall computational cost is linear in T, the expected cost per iteration must be constant. The proof gives a form for the overhead in terms of the operations in Algorithm 8. In practice this cost is observed to be negligible relative to the cost of step 2 of Algorithm 7, namely that of maximising $Q_n(\mu)$.

4.2.2.2 Step 2 : maximisation

To implement Step 2 of Algorithm 7 we first use the trivial observation that if $x_n > 0$ then $Q_n(\mu) < Q_{n-1}(\mu)$ for all $\mu < 0$. Thus to check if $\max_\mu Q_n(\mu) \geq \lambda$ we need only check this for $\mu > 0$. Similarly if $x_n < 0$ then we need only check for $\mu < 0$. To perform the check we just loop over all quadratics stored for either $\mu > 0$ or $\mu < 0$, and for each one check if its maximum is greater than λ. For a quadratic with stored triplet (τ, s, l) this involves checking whether

$$(S_n - s)^2 \geq 2\lambda(n - \tau). \tag{4.10}$$

If we flag a change at time n, then we can also output the value of τ corresponding to the quadratic whose maximum is largest, and this will be an estimate of the time of the change. The computational cost is thus proportional to number of quadratics that are stored, and can be bounded using the following result.

Theorem 4 *Let $x_1, \ldots, x_T, \ldots$ be a realization of the process $X_i = \mu_i + \epsilon_i$ where ϵ_i are independent, identically distributed continuous random variables with mean 0. Let the number of quadratics stored by FOCuS^0 for $\mu > 0$ at iteration T be $\#\mathcal{I}^0_{1:T}$. Then if μ_i is constant*

$$E(\#\mathcal{I}^0_{1:T}) \leq (\log(T) + 1),$$

while if μ_i has one change prior to T then

$$E(\#\mathcal{I}^0_{1:T}) \leq 2(\log(T/2) + 1).$$

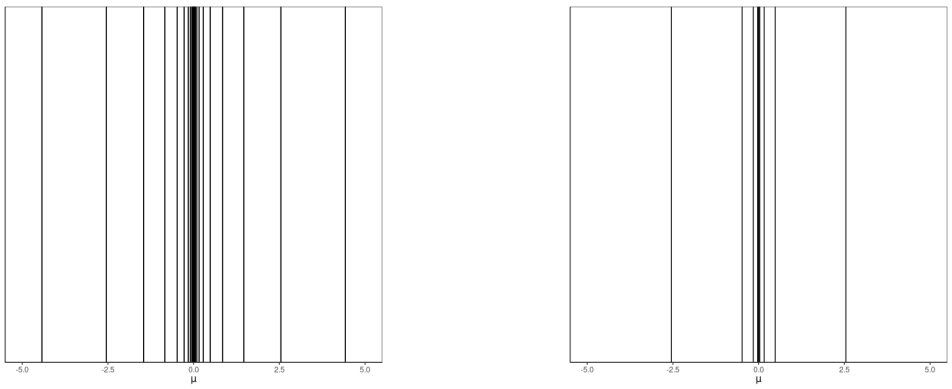
The proof for this and can be found in Appendix B.3. By symmetry, the same result holds for the number of quadratics stored for $\mu < 0$. The conditions of the data generating mechanism are weak – as the distribution of the noise can be any continuous distribution providing the noise is independent.

The theorem shows that the expected per-iteration time and memory complexity of FOCuS^0 at time T is $O(\log T)$. Furthermore, the expected per-iteration cost is essentially equal to checking (4.10) $\log(n) + 1$ times if there has not been a change, and for $2(\log(n/2) + 1)$ if there has been an, as yet, undetected change. A change of fixed size is detected in $O(1)$ iterations, and thus the overall computational time, for large T, will be dominated by the cost of iterations prior to the changepoint. For data for size one million, the bound on the number of quadratics is less than 15.

The FOCuS^0 algorithm is not strictly online, due to the cost per iteration not being bounded. But it is simple to introduce a minor approximation that is online. Assume we have a constraint that means we can find the maximum of at most P quadratics per iteration. A simple approximation is to introduce grid of points $\pm m_p$ for $m_p \in \mathbb{R}^+,\ p = 1, ...P$. There are then two natural approaches. One is that if we have $P + 1$ quadratics stored we pruned to P quadratics by removing the first quadratic whose interval does not contain a grid point. Alternatively we can keep all quadratics but only find the maximum of the quadratics whose interval contains a grid point. The advantage of this latter approach is that it avoids any approximations to $Q_n(\mu)$ which could propagate to future values of $Q_t(\mu)$ for $t > n$. Both these methods would dominate using the sequential-Page approach that used the same grid for μ_1 values. For example, if $\tilde{Q}_n(\mu)$ denotes the approximation to $Q_n(\mu)$ using the first approach, then we have $\tilde{Q}_n(\mu_1) = \mathcal{Q}_{n,\mu_1}$ for all μ_1 in our grid.

4.2.3 Simulation Study

We compare the FOCuS procedure with MOSUM, from (4.2) and the sequential Page-CUSUM statistics, from (4.6). In particular, the Page-CUSUM statistics was evaluated on grid of 25 values illustrated in Figure 4.3: the geometric grid is denser at the center in order to maximise power over changes of smaller magnitudes; similarly, MOSUM was evaluated over an equivalent set of window sizes such that $w_i = (\lambda/\mu_{1,i})^2$ with $\mu_{1,i}$ being a Page-CUSUM grid point. We furthermore compare the FOCuS^0 procedure with its approximation on a grid of 10 points, obtained from a subset of the same grid. This is roughly equivalent the number of intervals stored in FOCuS^0 over a sequence of 10 thousand observations where no change is present (as $\sum_{t=1}^{1\times10^5} 1/t \approx 9.789$). In Figure 4.4 we find a comparison of the maximum of the FOCuS^0 statistic against the values of the Page-CUSUM statistic evaluated on the super mentioned grid. We note that as the true post-change mean μ_1 falls exactly on one of the grid points of the Page-CUSUM statistics, then the value of the statistics for FOCuS^0 and



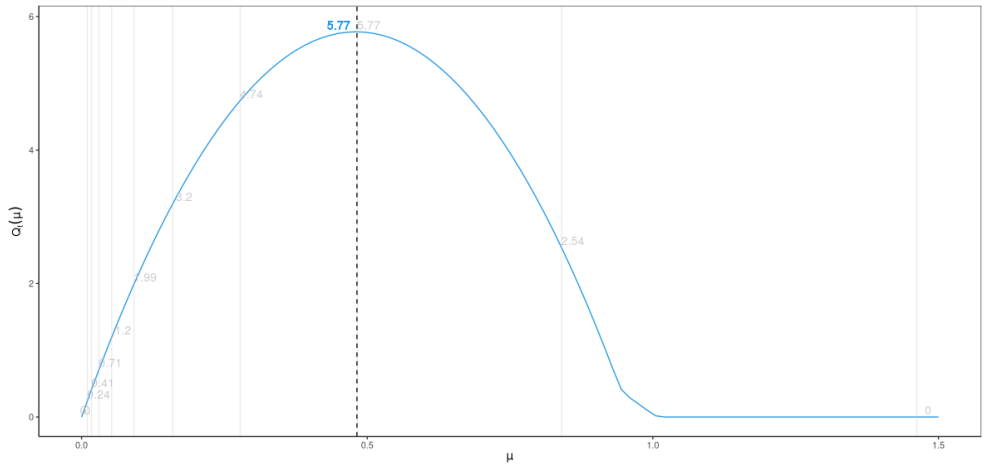
(a) Page-CUSUM (b) FOCuS^0 approximation

Figure 4.3: On the left, the 25 points grid employed for the simulations concerning the Page-CUSUM statitics, on the right, a subset of 10 grid points from the same grid, needed for the relative FOCuS approximation.

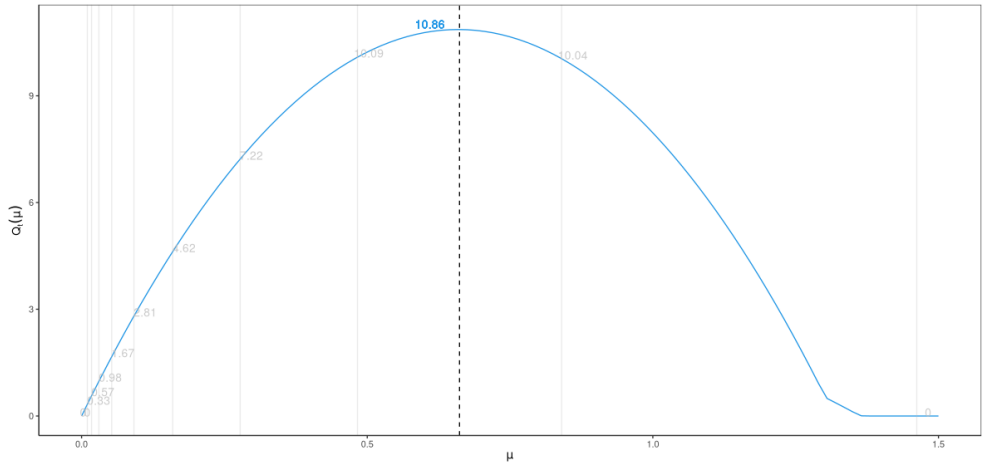
Page are going to be equivalent. However, as the post-change mean μ_1 falls exactly in the middle of two grid points, then the FOCuS statistics is going to be higher the Page-CUSUM statistic. Intuitively, Page-CUSUM should have an advantage whether the true change does fall exactly on one grid point, whereas in other situations FOCuS will perform better. On this observation, we build the following simulation study: we first evaluate the average run length in function of various values of a fixed penalty. Then we choose the smallest possible penalty for a guaranteed average run length, and we evaluate the performances of a method through the detection delay.

We evaluate the run length in function of a fixed threshold up to two million observations under the null $N(0, 1)$. Stopping times were recorded for a given threshold, and results were averaged across 100 different replicates. Results are reported in Figure 4.5. We notice how, up to a fixed run length, MOSUM threshold is lower then both FOCuS and Page-CUSUM. When comparing FOCuS with Page-CUSUM, of the benefits of the latter is in the slightly smaller threshold – this is expected as we have already denoted how the FOCuS^0 statistics will always dominate Page.

Fixed the thresholds as above, we compare the detection delay of the various implementations. We superimpose on the previously generated sequences a piecewise constant signal with a change at 1×10^5, and we evaluate the average detection delay by averaging the stopping time minus the real change location. The experiment is then repeated for a range of change magnitudes, among those we find all the Page-CUSUM grid points. Results are summarised in Table 4.1. We denote how FOCuS^0 shows a faster detection delay compared to the Page-CUSUM on changes of smaller magnitudes < 0.05. On larger magnitudes, Page-CUSUM slightly outperforms



(a)



(b)

Figure 4.4: A comparison of the FOCuS^0 cost (in blue) against the evaluation of the Page-CUSUM statistics on the grid introduced before (grey vertical lines) for two sequences with a change of magnitude 0.48, in (a), and of 0.66, in (b). The dashed black line refers to the true post-change mean. Blue labels refer to the maximum achieved by FOCuS^0 , grey labels to the value of the Page-CUSUM for each corresponding grid point in both cases.

FOCuS whether the post change mean falls in proximity of a grid point – this is because, again, at the grid points, both statistics have the same value, but Page's smaller threshold results in a faster detection. On the contrary, in between points, FOCuS outperforms Page, and this effect becomes negligible for changes of larger magnitudes. The FOCuS^0 approximation suffers of a slower detection delay when

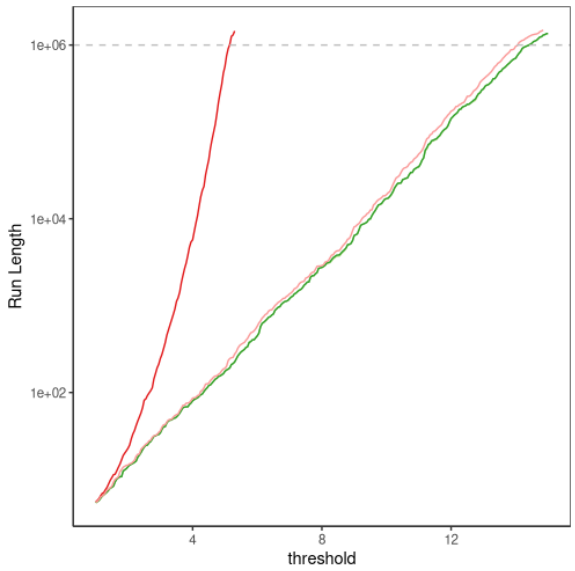


Figure 4.5: Average Run Length up to 1×10^6 in function of a fixed threshold for FOCuS^0 , in green, for Page-CUSUM, in light green and for MOSUM, in red. Log scale on the y axis.

compared to both Page-CUSUM and FOCuS^0 , however this is again reduced for larger magnitudes, and the method still outperforms MOSUM.

In Figure 4.4 we find a comparison of the FOCuS^0 cost function and the corresponding Page-CUSUM statistic evaluated on the various grid points. From that it is possible to notice that as the true post-change mean μ_1 falls exactly on one of the grid points of the Page-CUSUM statistics, then FOCuS and Page are going to be equivalent. However, in the case we have a post-change mean μ_1 exactly in the middle of two grid points, then the FOCuS statistics is going to outperform the Page-CUSUM statistic. This difference is less significant as the size of the change increases, and this explains why performances amongst the two implementations are similar for changes of larger magnitudes.

4.3 Extensions of FOCuS

In this section we discuss how the functional recursion of (4.7) can be extended to other models. We first show that Algorithm 8 also applies for the Gaussian unknown pre-change mean model and we recover similar guarantees in terms of average runtime complexity. For more complex models, we do not get such theoretical guarantees, but functional pruning techniques developed in Rigaill (2015); Maidstone et al. (2017)

	Detection Delay			
Magnitude	Page-CUSUM	FOCuS^0	FOCuS^0 10p	MOSUM
0.010	327399	**325288**	327468	392577
0.017	105578	**104713**	105743	126189
0.030	34426	**34069**	34210	38879
0.050	**12173**	12326	12683	14270
0.053	11109	**11088**	11445	12833
0.070	6568	**6495**	6704	7666
0.092	**4016**	4019	4139	4692
0.100	3371	**3371**	3460	3857
0.159	**1247**	1264	1283	1518
0.200	**800**	815	818	916
0.250	513	**510**	512	598
0.278	**410**	417	417	465
0.300	357	**353**	355	407
0.400	208	**207**	207	245
0.483	**141**	**141**	141	164
0.500	133	**132**	**132**	151
0.600	**95.2**	**95.3**	95.6	110
0.700	**70**	**70**	70.3	82.5
0.800	**51.9**	52.5	53.4	60.5
0.840	**48.2**	48.3	48.8	54.2
0.900	**42.8**	**42.8**	43.4	47.5
1.000	35.2	**35**	35.8	38.6

Table 4.1: Detection delays for Page-CUSUM, FOCuS^0 , and the FOCuS^0 approximation on a 10 points grid for 20 change magnitudes. In particular, on the left most column, we underline the magnitudes that fall exactly on the Page-CUSUM gridpoints.

apply. We illustrate that they are empirically efficient (in $O(\log(T))$ per iteration) for a loss function robust to outliers proposed in Fearnhead and Rigaill (2019).

4.3.1 FOCuS when the pre-change mean is unknown

Assume we are observing a sequence of observations $x_1, \ldots, x_n$ distributed as a $N(\mu_0, \sigma)$ under the null and as $N(\mu_1, \sigma)$ under the alternative, with σ known and $\mu_1 \neq \mu_0$. As suggested by Yu et al. (2020), a natural test for a change in the likelihood ration statistic

$$\mathcal{Q}_n = \max_{\substack{\tau \in \{1,\ldots,n\} \\ \mu_0, \mu_1 \in \mathbb{R}}} \left\{ \sum_{t=1}^{\tau} (x_t - \mu_0)^2 - \sum_{t=\tau+1}^{n} (x_t - \mu_1)^2 \right\}. \tag{4.11}$$

Yu et al. (2020) present finite-sample results that demonstrate the statistical optimality of such a test. They also present algorithms for evaluating this test statistic. The fastest algorithm that avoids any approximation is $O(n)$ in computational complexity per iteration while being $O(n)$ in storage, which make their methodology

infeasible to a true online setting. For the rest of this paper will refer to such algorithm as Yu-CUSUM.

Let us explain how we can solve (4.11) using the functional recursion of FOCuS^0 . A key observation is that the difference in cost between a change at τ and a change at n for any two means μ_0 and μ_1 can be written as:

$$\sum_{t=1}^{\tau}(x_t-\mu_0)^2+\sum_{t=\tau+1}^{n}(x_t-\mu_1)^2-\sum_{t=1}^{n}(x_t-\mu_0)^2 = -(\mu_1-\mu_0)\left(2\sum_{\tau+1}^{n}x_t-\mu_0-\mu_1\right) \quad (4.12)$$

The right hand size can be rewritten as $-\delta(2\sum_{\tau+1}^{n}x_t - m)$ taking $\delta = \mu_1 - \mu_0$ and $m = \mu_0 + \mu_1$. As before we can consider separately the case $\delta > 0$ (up-change) and $\delta < 0$ (down-change). We will discuss the case $\delta > 0$ only, as the other case follows immediately by symmetry. For $\delta > 0$ the sign of (4.12) only depends on m, and we recover, as in (4.8) for the known pre-change mean case, that a change at n is better than a change at τ if m is in the interval :

$$\left[2\sum_{t=\tau+1}^{n}\frac{x_t}{(n-\tau)}\ ,\ +\infty\right). \quad (4.13)$$

Hence, we can update the intervals and quadratics corresponding to candidate changes exactly as in Step 1 of FOCuS^0 using Algorithm 8.

We provide in Appendix B.2 a pseudo-code description of the FOCuS algorithm in case the pre-change mean is not known, but in essence there are only two small differences between FOCuS and FOCuS^0 :

1. For the interval update (step 1), in FOCuS^0 for up-changes we could restrict our attention to $\mu_1 \in [\mu_0, +\infty)$ (resp. $(-\infty, \mu_0]$ for down-changes), whereas in FOCuS, not knowing the value of the first segment mean, we need to consider all cases for m, i.e. $m \in (-\infty, +\infty)$.

2. For the maximisation (step 2) in FOCuS^0 we only need to optimize the value of the last segment, whereas in FOCuS we also need to optimize the pre-change mean.

We derive in Theorem 5 in appendix B.3 the same bound as FOCuS^0 on the expected number of candidates showing that the expected per-iteration time and memory complexity of FOCuS at time T is $O(\log(T))$.

4.3.2 FOCuS in the presence of outliers

Further extensions of FOCuS are to use different loss functions to the square error loss obtained from a Gaussian log-likelihood. Motivated by the application in Section 4.4

we will consider a robust loss function, the biweight loss, which enables us to detect changepoints in the presence of outliers (see Fearnhead and Rigaill, 2019). We define thie loss as

$$L(x_t, \mu_1) = -\max\left\{\left(\frac{\mu_1}{2} - x_t\right)^2, K\right\}, \tag{4.14}$$

where K is a user specified threshold. Using that in our online setting we recover the following functional recursion:

$$Q_n(\mu) = \max\left\{\sum_{t=1}^{n} L(x_t, 0),\ Q_{n-1}(\mu) + L(x_n, \mu)\right\}. \tag{4.15}$$

Using ideas described in Section 3.2 of Fearnhead and Rigaill (2019) it is straightforward to implement this recursion for all μ efficiently. For this model we are unable to recover a bound on the expected number of candidate changepoints. However we observed empirically that the cost for iteration T is in $O(\log(T))$ (see Figure 4.6).

4.3.3 Simulation Study

In Figure 4.6 we find a comparison of the runtime between FOCuS^0 , FOCuS , the robust implementation introduced in (4.15) (R-FOCuS) and Algorithm 3 from Yu et al. (2020), denoted as Yu-CUSUM. Runtimes were recorded for multiple finite sequences of lengths ranging from 100 to 5×10^4. To produce a fair comparison both implementations were written in C++, all simulations were performed on a common personal computer. We find little difference when comparing FOCuS^0 with FOCuS , both showing an empirical linear increase in timings with the latter being slightly slower. When comparing FOCuS to Yu-CUSUM, we find a comparable runtime only up to $n = 100$, after which FOCuS is generally faster, in particular on larger sequences, given that Yu-CUSUM shows quadratic complexity. Lastly, we notice how R-FOCUS, while still retaining a linear computational complexity, has a larger overhead compared to the simpler implementations.

As one could estimate the mean of a Gaussian process when the pre-change mean is unknown, and use that value to run the algorithms introduced in Section 4.2, the second comparison we make is between FOCuS^0 with the pre-change-mean known learned over a training sequence and FOCuS with the pre-change mean unknown. We study in particular the performances of FOCuS^0 as we vary the size of training data from 100 observations up to 1×10^5. We compare both average run-length as a function of the threshold, and detection delay as a function of the magnitude of a change. For each experiment, we report summaries over 100 replicates, and the results on detection delay are for thresholds chosen so each algorithm has an average run-length of 1×10^6. In all cases we simulate data with 1×10^5 data points prior to the change. Results are summarised in Figure 4.7.

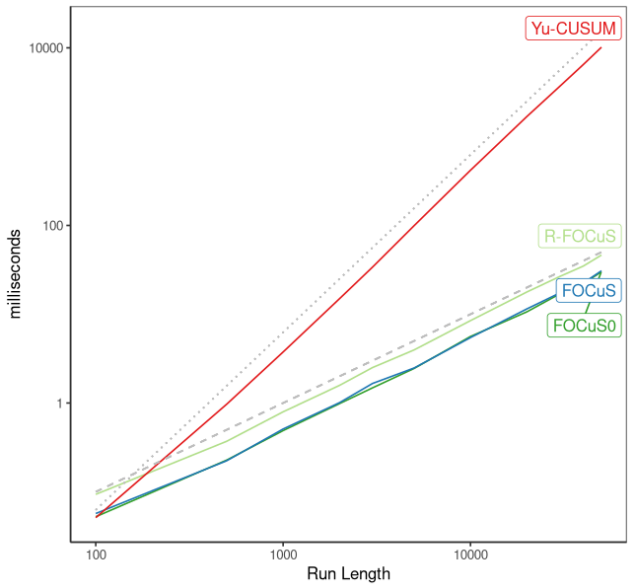
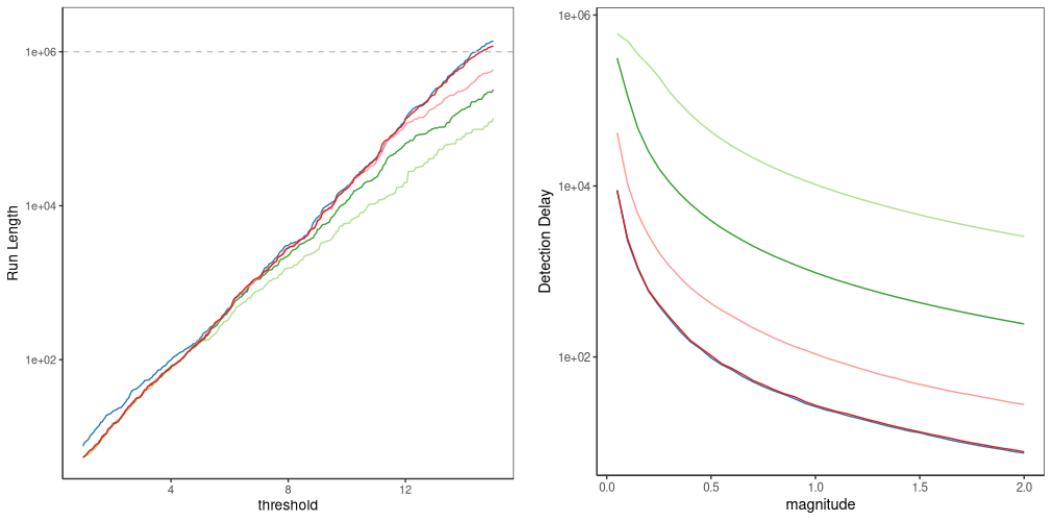


Figure 4.6: Runtime in milliseconds of FOCuS^0 , FOCuS , R-FOCuS and Yu-CUSUM in function of the length of the sequence (log-scale on both axes). Grey lines refer to an expected $\mathcal{O}(n)$ increase (dashed) and $\mathcal{O}(n^2)$ increase (dotted).

FOCuS with pre-change out performs FOCuS^0 . In part this is because it requires a smaller threshold to achieve the same average run length. Furthermore the detection delay of FOCuS^0 can be substantially increased if the estimate of the pre-change mean is close to post-change mean; whereas the reduction in detection delay when the estimate of the pre-change mean is away from the post-change mean is much less. The advantage of FOCuS is that it can improve its estimate of the pre-change mean using the data prior to any change. Thus we see substantial benefits of FOCuS relative to FOCuS^0 when the amount of training data is small. It is only when the amount of training data is of the same order as the amount of data prior to the change that FOCuS^0 gives similar results to FOCuS .

4.4 Application of FOCuS to the AWS Cloudwatch CPU utilization

We now evaluate FOCuS by comparing with a bespoke anomaly detection algorithm on the Amazon CPU utilization datasets from the the Numenta Anomaly Benchmark (Ahmad et al., 2017). The aim with these datasets is to detect anomalous behaviours in the CPU utilization of various Amazon Cloudwatch instances. For each dataset anomalous behaviours have been manually flagged by experts, and those stand as the



(a) Average run length in function of a fixed threshold.

(b) Detection delay in function of various magnitudes of a change.

Figure 4.7: Comparison between FOCuS pre-change unknown and pre-change known. Blue line corresponds to FOCuS with pre-change unknown, while the other lines correspond to FOCuS^0 for different training set sizes: 100 (light green); 1000 (dark green); 1×10^4 (pink); and 1×10^5 red. Both figures have a log-scale on the y axis.

ground truth. The data sets are shown in Figure 4.8, and demonstrate a range of behaviour. As point anomalies are common we will use the R-FOCuS algorithm.

When evaluating algorithms we will follow the methodology in Ahmad et al. (2017). A detection is deemed to be correct if it lies within $\pm 0.05 \cdot n$ of the true anomaly, where n is the length of the time series; and multiple detection within the window are allowed. A method can use the first 15% of each dataset, a portion of data known to not include any anomalies, to set tuning parameters. We use this data to tune both K in the biweight loss and the detection threshold as described in Appendix B.4.

As some data sets have multiple anomalies to be detected, we have to adapt R-FOCuS so that it does not stop once a change to some anomalous behaviour has occurred. To adapt R-FOCuS we simply initiate the procedure again at the estimated changepoint location after a detection is triggered. In order to reduce the number of false positives and to extend the average run length of the algorithm, at each detection we inflate the penalty by a factor of $\log(\tau_s)/\log(\tau_s - \tau_{s-1})$, with $\tau_0, ..., \tau_k$ being a vector of estimated changepoint locations.

We compare R-FOCuS with numenta HTM, the best performing algorithm to date on these data. Numenta HTM (Ahmad et al., 2017) is an anomaly detection

algorithm that employs an unsupervised neural network model to work with temporal data (Cui et al., 2016) to perform anomaly detection.

Results are summarised in Figure 4.8 and Table 4.2. We find that R-FOCuS has better performances in term of Precision, the proportion of true anomalies detected, and Recall, the proportion of detections that are true anomalies, compared with Numenta HTM. On a case to case basis, in most of the sequences both algorithm flagged correctly the anomalous behaviours. HTM overall achieves slightly shorter detection delays (with the exception of **f**), however it produces more false positives (13 false detections against 7 of R-FOCuS). In terms of missed detections, both algorithms perform similarly, with R-FOCuS missing and anomaly in **a** which flagged by HTM, whilst in **d** we observe the opposite.

Overall, this shows the flexibility of R-FOCuS for online change detection, especially considering that R-FOCuS is a simpler approach which is operating under model misspecification, and with significantly shorter computational run-times than Numenta HTM (on 3000 observations R-FOCuS takes roughly 2 milliseconds against the 4 minutes for HTM).

Detector	Precision	Recall
R-FOCuS	0.58	0.82
Numenta HTM	0.50	0.76

Table 4.2: Precision and Recall for R-FOCuS and Numenta HTM.

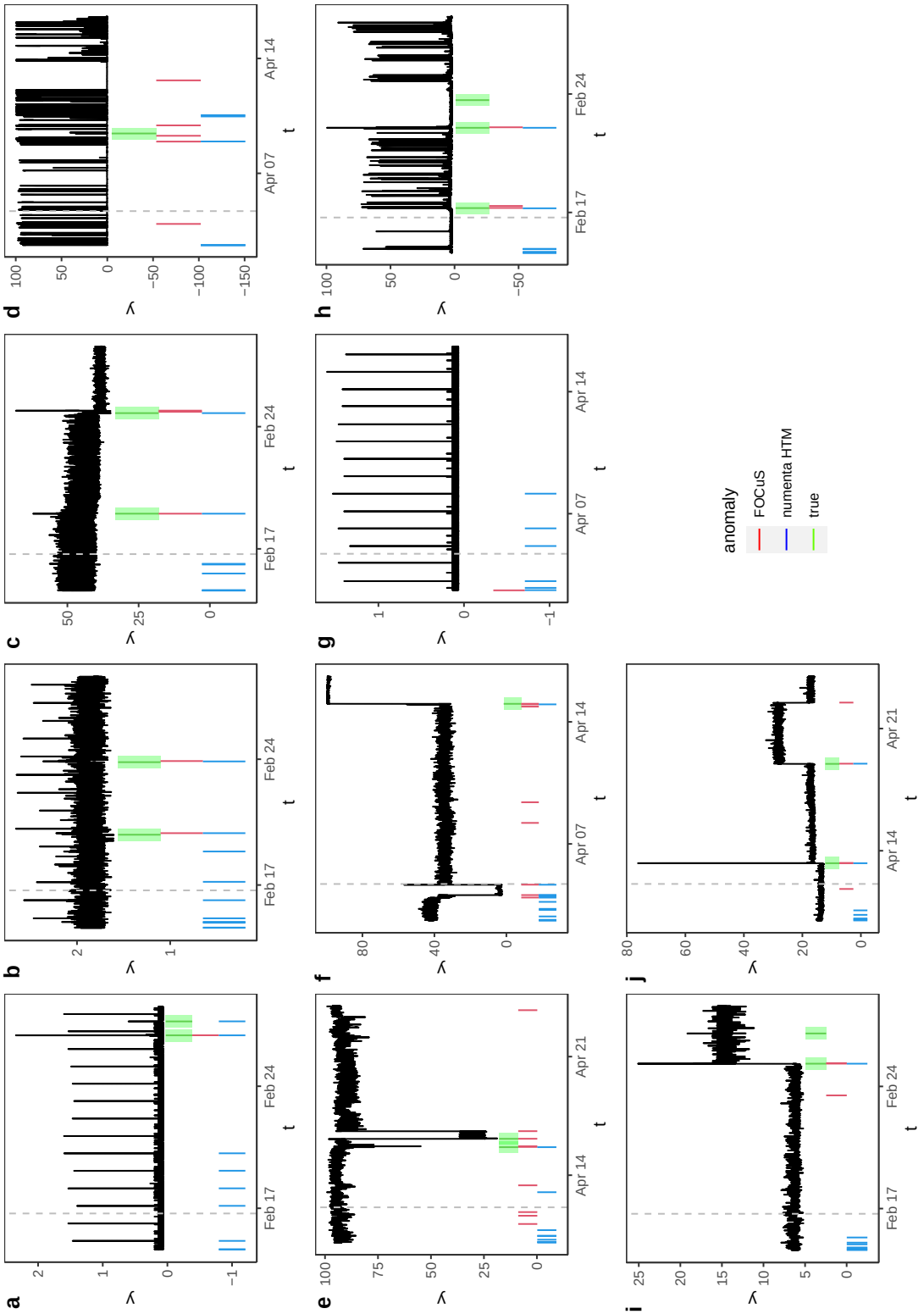


Figure 4.8: The 8 (**a** - **j**) different time series of AWS Cloudwatch CPU utilization. Green, red and blue segments correspond, respectively, to the real and estimated anomaly locations of R-FOCuS and Numenta HTM. The green rectangle around each anomaly is the anomaly window (the area in which an anomaly must be detected to count as a true positive). Multiple detections within the green lines are allowed, and are not considered as false positives. The dashed line corresponds to the probation period, used for training of tuning parameters: detections before the dashed line are not accounted for in the final result.

Chapter 5

A Nonparametric Approach to Online Anomaly Detection

5.1 Introduction

The challenge of sequential nonparametric changepoint detection has seen significant development in recent years. See, for example, Tartakovsky et al. (2014) for an excellent introduction to the area. Contributions include the work of Gordon and Pollak (1994), Ross and Adams (2012), and Padilla et al. (2019), for example, who introduce novel approaches to detect changes in an unknown distribution. Others, including Chakraborti and van de Wiel (2008); Hawkins and Deng (2010); Murakami and Matsuki (2010); Ross et al. (2011); Mukherjee and Chakraborti (2012); Liu et al. (2013); Wang et al. (2017) and Coelho et al. (2017) seek to address a different nonparametric challenge: the sequential detection of changes in the mean, scale, or the location of the data. Such methods have also found application in a range of fields including monitoring financial systems (Pepelyshev and Polunchenko, 2017), monitoring viral intrusion in computer networks (Tartakovsky and Rozovskiĭ, 2007), detecting changes in social networks (Chen, 2019), genome sequencing (Siegmund, 2013), and radiological data (Padilla et al., 2019).

Our work is motivated by novel challenges increasingly encountered within many contemporary digital settings, such as those found in the telecommunications sector. In such environments it is increasingly important to perform device-side analyses on units with limited computational power and data storage capability or adding as little computational overhead as possible. Existing methods, such as those mentioned above are unsuitable for use in such cases as they require the entire data stream to be stored and analysed *a posteriori*, whereas our memory-constrained setting makes this impossible. As a consequence, an online approach that uses a lighter data footprint is required.

The first example we present, encountered by an industrial collaborator, can be seen in Figure 5.1. Here we display two sample data sets of a key operational metric that are routinely monitored to identify problems with networking devices. Figure 5.1(a) displays data from a healthy device, whereas the data in Figure 5.1(c) contains an event that triggers a user intervention. Note, in particular how the structure of the data changes in the region when the event occurs. This can perhaps be more clearly seen in Figures 5.1(b,d). The ideal, therefore, is to be able to (i) identify the start of changing structure in advance of the user being required to start an intervention – we call the correct detection of such an event an 'anticipation'; (ii) using an approach that does not necessarily require the same underlying distribution pre- and post-change and (iii) can still permit (more subtle) non-anomalous changes in structure that occur over time due to typical operational issues (e.g. electrical interference, line optimisation etc).

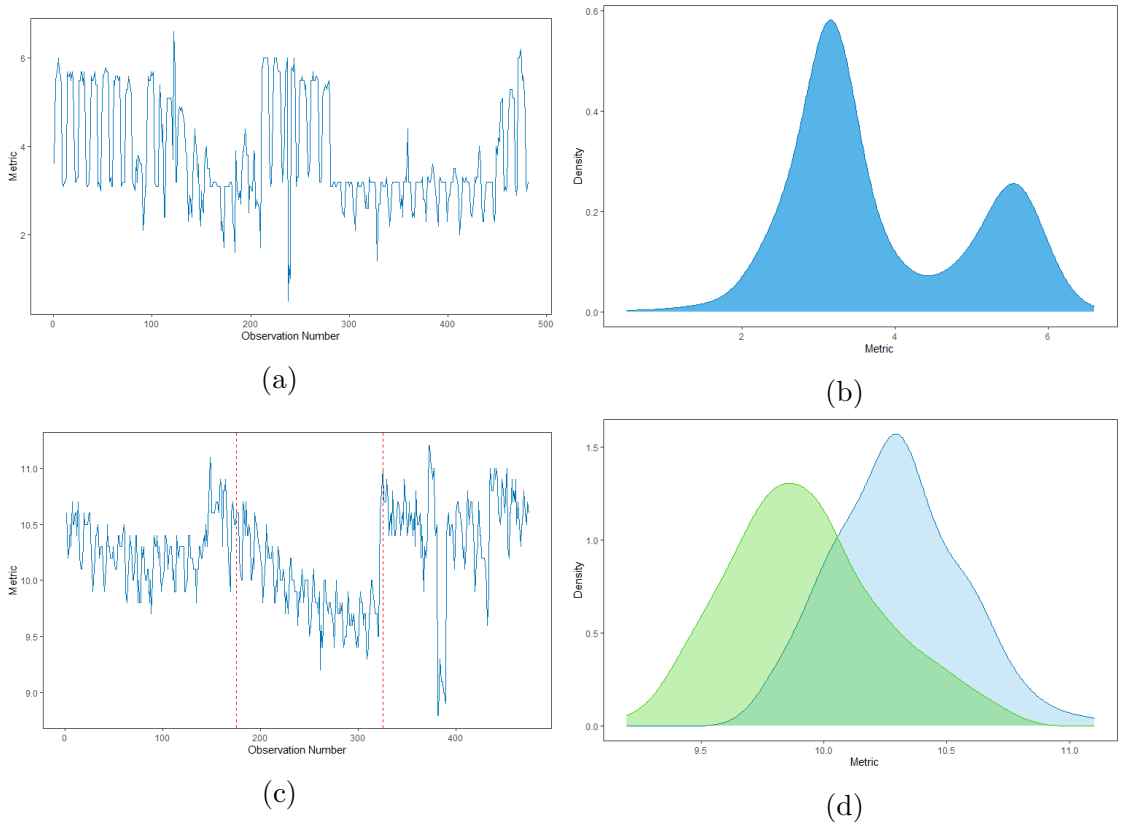


Figure 5.1: Example of telecoms operational data: (a) a series without an event, and (c) a series with an event taking place between the two red lines. The corresponding kernel density estimates are presented in (b) and (d) respectively.

The second example aims at monitoring for the movement of a DualShock

controller through the y-axis reading from the accelerometer. Each data stream consists of 2000 observations, over 100 different data series. The controller is then directly or indirectly moved at a known time, changing its state. In Figure 5.2 we present an example of each of the four different types of movement considered, and remark that we model the movement of the controller as a change in distribution. The aim of this application is to detect the presence of an user trough the movement of the controller as quickly as possible. An online efficient method is required due to the high frequency at which the data is recorded and not to have a huge computational overhead.

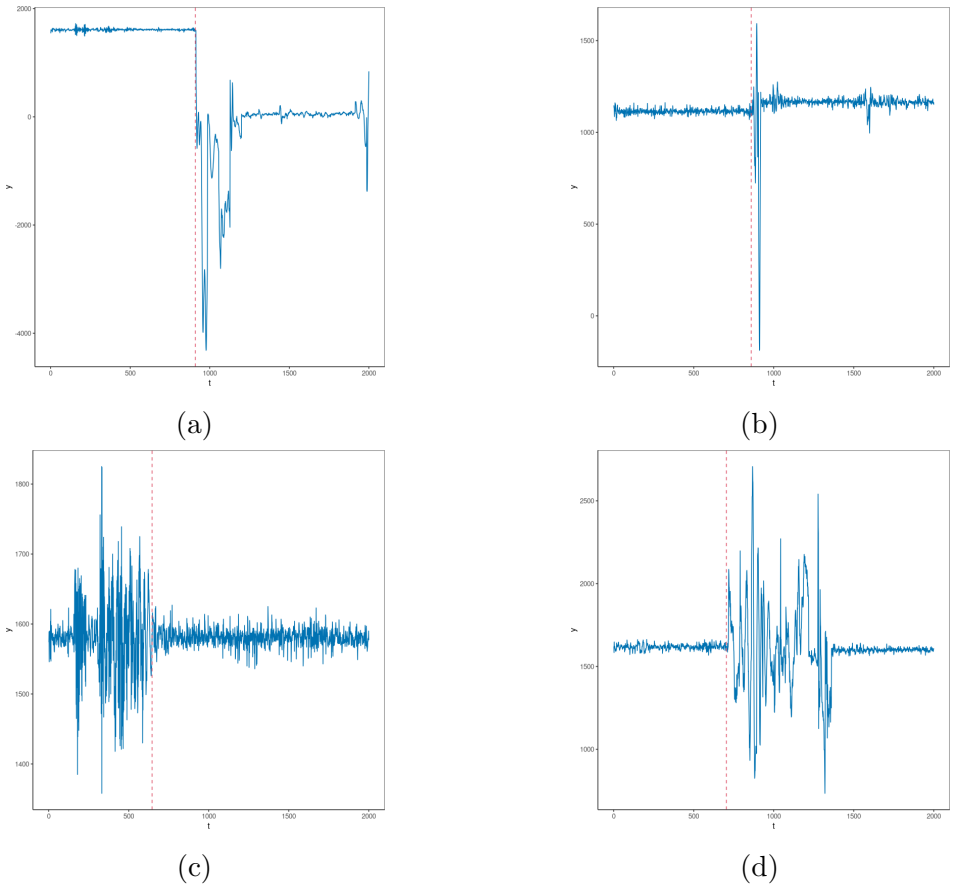


Figure 5.2: Examples of the four different types of movement experienced by the controller. These correspond to picking up the controller (a), sitting on a sofa where the controller is lying (b), shaking the controller (c), and sliding the controller along a table (d).

Many existing methods, such as those mentioned above, are unsuitable for use in this setting as they typically require the entire data stream to be stored.

Unfortunately in our problem setting, such memory constraints are no longer possible. To overcome this one might, for example, consider adopting an online nonparametric changepoint detection approach using a sliding window, such as in the MOSUM test (Chu et al., 1995; Eichinger and Kirch, 2018; Kirch et al., 2018; Meier et al., 2021). Alternatively a control chart based approach, e.g. Ross and Adams (2012), might be considered. Unfortunately, as described above, our telecoms operational metric can exhibit non-anomalous shifts in mean and variance. Such structure, while operationally acceptable, may cause a control chart based approach to return excessive false alarms due to the cumulative nature of the test statistic. Consequently we choose to adopt a sliding window to guard against this. Further, existing nonparametric sequential changepoint detection methods prove unsuitable as they do not expect the null distribution to change over time. To this end we introduce a new windowed, nonparametric procedure to detect sequential changes in an online setting. Taken from the Latin, *nunc* ('now'), our approach provides a **N**onparametric **UN**bounded **C**hangepoint (NUNC) detection.

We propose two variants of NUNC: NUNC Local, and NUNC Global. The first of these algorithms, NUNC Local, performs the detection in a sliding window, considering only the points inside this window. This allows for the implementation to work in an online setting. The second, NUNC Global, uses an efficient updating step to compare the distribution of the historic data seen with the distribution of the data inside the sliding window; if these differ significantly, a change is identified. The rest of this paper is organised as follows: In Section 5.2 we outline the methodology behind our new sequential tests. In particular, we detail the existing nonparametric changepoint methods our work is based upon, and in Section 5.2.1 we provide details of our two new window-based changepoint detection tests. The remainder of the section then explores the properties of the test, including the choice of quantiles and threshold in Section 5.2.2. We then explore the performance of NUNC Local and NUNC Global using both simulated scenarios (Section 5.3) and data arising from the previously described telecommunications setting (Section 5.4).

5.2 Background and Methodology

Our approach builds on the recent work of Zou et al. (2014) and Haynes et al. (2017a), utilising a nonparametric likelihood ratio test as the basis for the proposed sequential Nonparametric test. In so doing, the method permits a range of data distributions to be modelled, without the need for restrictive parametric assumptions. Below we introduce both NUNC approaches, and provide a discussion of their various features including computational performance and the choice of quantiles. However, prior to doing so, we review the pertinent literature on nonparametric changepoint methods.

We begin by outlining some notation. Assume that we observe a data stream

of real valued independent observations $x_1, x_2, \ldots, x_t$, and that the data stream can contain a changepoint, at some (unknown) time point τ. Further, assume that x_1 is the start of this data stream, x_t is the most recently observed point, and that for $j > i$, $x_{i:j} = x_i \ldots, x_j$ denotes a segment of the data. If a change is present in the data at τ_1, then we refer to $x_1, \ldots, x_{\tau_1}$ as the pre-change segment, drawn from a distribution $F_1(\cdot)$. Similarly, $x_{\tau_1+1}, \ldots x_{\tau_2}$ are considered to be drawn from post-change distribution, $F_2(\cdot)$, with $F_1 \neq F_2$.

Following Zou et al. (2014), let $F_{i:t}(q)$ denote the (unknown) cumulative distribution function (CDF) for the segment $x_i, \ldots, x_t$, and $\hat{F}_{1:t}(q)$ as its associated empirical CDF. I.e.

$$\hat{F}_{1:t}(q) = \frac{1}{t}\left\{\sum_{j=1}^{t} \mathbb{I}(x_j < q) + 0.5 \times \mathbb{I}(x_j = q)\right\}. \tag{5.1}$$

Under the assumption that the data are independent, then the empirical CDF will follow a Binomial distribution. That is,

$$t\hat{F}_{1:t}(q) \sim \text{Binom}(t, F_{1:t}(q)). \tag{5.2}$$

Using the Binomial distribution, we write the log-likelihood of the segment $x_{\tau_1+1}, \ldots, x_{\tau_2}$ as

$$\mathcal{L}(x_{\tau_1+1:\tau_2}; q) = (\tau_2 - \tau_1)\left[\hat{F}_{\tau_1+1:\tau_2}(q)\log(\hat{F}_{\tau_1+1:\tau_2}(q)) - (1 - \hat{F}_{\tau_1+1:\tau_2}(q))\log(1 - \hat{F}_{\tau_1+1:\tau_2}(q))\right]. \tag{5.3}$$

Consequently, equation (5.3) can be used to form a likelihood ratio test statistic for the detection of a change at a single quantile of the distribution as follows:

$$\max_{1\leq\tau\leq t} \mathcal{L}(x_{1:\tau}; q) + \mathcal{L}(x_{\tau+1:t}; q) - \mathcal{L}(x_{1:t}; q).$$

Following Zou et al. (2014) and Haynes et al. (2017a), this test statistic can be averaged over multiple quantiles, $q_1, \ldots, q_K$, in order to search for a change in distribution. The statistic for such a test can be formulated as follows:

$$C_K(x_{1:t}) = \max_{1\leq\tau\leq t} \frac{1}{K}\sum_{k=1}^{K} 2\left[\mathcal{L}(x_{1:\tau}; q_k) + \mathcal{L}(x_{\tau+1:t}; q_k) - \mathcal{L}(x_{1:t}; q_k)\right]. \tag{5.4}$$

Here K is the fixed number of quantiles to be averaged over. Haynes et al. (2017a) propose that a value of K is chosen that is proportionate to $\log(t)$. The choice of quantiles at which the empirical CDF can be evaluated will be discussed later in Section 5.2.2.

Using this test, a changepoint is declared when $C_K(x_{1:t}) - \beta \geq 0$. Thus the stopping time for our test becomes

$$\max_{1\leq\tau\leq t} \sum_{k=1}^{K} 2\left[\mathcal{L}(x_{1:\tau}; q_k) + \mathcal{L}(x_{\tau+1:t}; q_k) - \mathcal{L}(x_{1:t}; q_k)\right] \geq K\beta, \tag{5.5}$$

where β is the threshold for the test.

Having outlined how the existing (offline) nonparametric tests work, and how the cost function from this work can be used to devise a stopping rule for a sequential changepoint detection test, we are now in a position to introduce our two variant nonparareteric approaches.

5.2.1 Two Sequential Changepoint Detection Algorithms

We now introduce two different, yet related, approaches that can be adopted within this nonparametric framework: NUNC Local and NUNC Global. Common to both is the use of a sliding window, and the test statistic given in equation (5.4). Where the two approaches differ, however, is the manner in which the data observed outside the window are handled. In NUNC Local, a simplistic perspective is adopted, taking the data contained within the sliding window into account – i.e., previously seen points that fall outside this window are forgotten. The advantage of this approach is that the sequential test is immune to false alarms that might be caused, for example, by a natural drift in the underlying distribution of the data. The drawback, however, is that the empirical CDF must be estimated only from the data in the window and so any historic information is lost.

NUNC Global seeks to overcome the short-comings of NUNC Local. Specifically, NUNC Global stores the empirical CDF that has been estimated using all data observed so far, and tests whether the data from such empirical distribution differ from the data observed in the current window. In Section 5.4 we will seek to contrast the differences between these to variants. However, prior to this, we describe both search methods more carefully, whilst also describing various properties and recommendations.

5.2.1.1 NUNC Local

Our first method takes a sliding window of size W and performs the test on the data within this sliding window. In the sliding window of points we have that

$$\mathcal{Q}_t^{local} = \max_{t-W+1 \leq \tau \leq t} \sum_{k=1}^{K} 2\left[\mathcal{L}(x_{t-W+1:\tau}; q_k) + \mathcal{L}(x_{\tau+1:t}; q_k) - \mathcal{L}(x_{t-W+1:t}; q_k)\right], \quad (5.6)$$

where K is the number of quantiles and β is the test threshold. When $\mathcal{Q}_t^{local} \geq K\beta$ then the algorithm stops at time t and declares that a change has occurred at time τ. A description of NUNC Local pseudocode can be found in Algorithm 9.

The choice of the parameters for NUNC Local, including the window size, quantiles, and threshold; will be discussed in Section 5.2.2 and in simulations in Section 5.3. We remark here, however, that the choice of K and the size of the

window is related to the computational cost of NUNC Local. In particular, this cost is $K \cdot \mathcal{O}(W^2)$.

Algorithm 9: NUNC Local Algorithm

Data: $\{x_{t-W+1}, ..., x_{t-1}, x_t\}$, the last W realizations from a data generating process X.

Input: $\beta > 0$, $K < W$, $q_1, \ldots, q_k$ quantiles

$c \leftarrow -log(2W - 1)$;

$\mathcal{Q} \leftarrow \max_{t-W+1 \leq \tau \leq t} \left[\sum_{k=1}^{K} 2 \left(\mathcal{L}(x_{(t-W):\tau}; q_k) + \mathcal{L}(x_{\tau+1:t}; q_k) - \mathcal{L}(x_{(t-W):t}; q_k) \right) \right]$;

$\tau^* \leftarrow \arg\max_{t-W+1 \leq \tau \leq t} \left[\sum_{k=1}^{K} 2 \left(\mathcal{L}(x_{(t-W):\tau}; q_k) + \mathcal{L}(x_{\tau+1:t}; q_k) - \mathcal{L}(x_{(t-W):t}; q_k) \right) \right]$;

if $\mathcal{Q} \geq K\beta$ **then**

  Return τ^* as a changepoint

end

In order to reduce the computational requirements of NUNC Local, which is quadratic in window size, it is possible to instead perform the search on a subset of the points in the sliding window. In this setting, we obtain the stopping condition:

$$\max_{\tau \in B_J} \sum_{k=1}^{K} 2 \left[\mathcal{L}(x_{t-W+1:\tau}; q_k) + \mathcal{L}(x_{\tau+1:t}; q_k) - \mathcal{L}(x_{t-W+1:t}; q_k) \right] \geq K\beta, \tag{5.7}$$

where $B_J \subset \{t - W + 1, \ldots, t\}$. This corresponds to changing the maximisation in Algorithm refalg: Forgetting to taking place over the set B_J rather than the entire window. Using a subset of size $J << W$, the computational cost of NUNC Approximate is reduced to $JK \cdot \mathcal{O}(W)$. To find a suitable subset of values to search inside the sliding window, we first note that intuitively it only makes sense to search for a change in the right hand half of the window. This is because the data in the left of the window has already been scanned for a change several times. Moreover, we can also establish the following: that there exists a point on the right hand side of the window such that a changepoint cannot be detected to the right of this point.

For any quantile q the test statistic is bounded such that

$$\mathcal{L}(x_{t-W+1:\tau}; q) + \mathcal{L}(x_{\tau+1:t}; q) - \mathcal{L}(x_{t-W+1:t}; q) \leq -\frac{\tau}{W} \log \frac{\tau}{W} - (W - \tau) \log \left(\frac{W - \tau}{W} \right). \tag{5.8}$$

Furthermore, for fixed W, this equation is decreasing as τ increases, and so if τ^* is the point such that

$$-\frac{\tau^*}{W} \log \frac{\tau^*}{W} - (W - \tau^*) \log \left(\frac{W - \tau^*}{W} \right) \leq \frac{\beta}{2}$$

then for $\tau > \tau^*$ detection of a change is impossible.

Proof 1 *See Appendix.*

As a consequence of Proposition 6, only a portion of the right hand side of the window needs to checked, and the value of this cutoff can be found, with the value for τ^* being calculated numerically. Further computational efficiencies can be realised for NUNC Local if it only performed on a spaced out grid of points. This is due to the value of the test statistic being correlated at nearby points. Consequently, if segmenting the data at t does not return a change, then it is unlikely that a change will be detected at $t+1$. As a result, we propose to use an equally spaced grid of J points starting from the centre of the window, after it has been trimmed using the value of τ^*. However, one drawback of using the grid method is that there will be a higher detection delay for smaller values of J. This is due to it taking longer for the change to reach a point that we are checking. As such, we conclude that there is a trade-off between computational efficiency and detection delay when using the approximated algorithm.

5.2.1.2 NUNC Global

NUNC Global differs from the NUNC Local. Specifically it tests whether or not the data in the window comes from a different distribution to all the data seen so far. To store the information in a memory efficient manner, we again fix K quantiles and update the longrun empirical CDF, denoted by $z_W^{(t)}(\cdot)$, each time a point leaves the sliding window. The recursive equations for this update step are as follows:

$$\begin{aligned} z_W^{(W)}(q) &= \hat{F}_{1:W}(q), \\ z_W^{(t+1)}(q) &= \frac{1}{t-W+1}\left[(t-W)z_W^{(t)}(q) + \hat{F}_{(t-W+1):(t-W+1)}(q)\right], \quad t \geq W. \end{aligned} \tag{5.9}$$

I.e. the long run empirical CDF is updated to take into account the point that will leave the sliding window at the next iteration. The Global algorithm then compares the distribution for the long run empirical CDF to the distribution of the data in the sliding window, denoted by $\hat{F}_{t-W+1:m}(\cdot)$.

To implement this approach, we need to obtain a CDF estimate of the full data. This is given by a weighted mixture of the long run empirical CDF and the current segment empirical CDF estimate. Assuming we are at time, m, and have a sliding window of size, W, we write this as

$$\hat{F}_{\text{full}}(q) = \hat{F}_{1:t}(q) = \frac{t-W}{t} z_W^{(t)}(q) + \frac{W}{t}\hat{F}_{t-W+1:t}(q).$$

With these distributions in place, we can obtain the equivalent likelihoods, given respectively by

$$\begin{aligned}
\mathcal{L}(x_{1:t-W}; t) &= (t-W)\left[z_W^{(t)}(q)\log(z_W^{(t)}(q)) - (1-z_W^{(t)}(q))\log(1-z_W^{(t)}(q))\right] \\
\mathcal{L}(x_{t-W+1:t}; t) &= W\left[\hat{F}_{t-W+1:t}(q)\log(\hat{F}_{t-W+1:t}(q)) - (1-F_{t-W+1:t}(q))\log(1-\hat{F}_{t-W+1:t}(q))\right] \\
\mathcal{L}(x_{1:t}; t) &= t\left[\hat{F}_{\text{full}}(q)\log(\hat{F}_{\text{full}}(q)) - (1-\hat{F}_{\text{full}}(q))\log(1-\hat{F}_{\text{full}}(q))\right]. \qquad (5.10)
\end{aligned}$$

The test statistic is then given by:

$$\mathcal{Q}_t^{global} = \sum_{k=1}^{K} 2\left[\mathcal{L}(x_{1:t-W}; q_k) + \mathcal{L}(x_{t-W+1:t}; q_k) - \mathcal{L}(x_{1:t}; q_k)\right]. \qquad (5.11)$$

When $\mathcal{Q}_t^{global} \geq K\beta$ we stop and declare a change at time t. Pseudocode outlining the Global algorithm is provided in Algorithm 10. We note that the computational cost of NUNC Global is $K\mathcal{O}(W)$.

Algorithm 10: NUNC Global Algorithm

Data: $x_{(t-W+1):W}$, the last W realizations from a data generating process X; $z_W^{(t)}(q_k)$ for $q_k \in t_{1:K}$

Input: $\beta > 0$; $K < W$; $t_{1:K}$ the fixed quantiles.

$\mathcal{Q} \leftarrow \sum_{k=1}^{K} 2\left[\mathcal{L}(x_{1:t-W}; q_k) + \mathcal{L}(x_{t-W+1:t}; q_k) - \mathcal{L}(x_{1:t}; q_k)\right]$;

if $\mathcal{Q} \geq K\beta$ **then**

  Return $t - W$ as a changepoint.

else

  $z_W^{(t+1)}(q_k) \leftarrow \frac{1}{t-W+1}\left[(t-W)z_W^{(t)}(q_k) + F_{(t-W+1):(t-W+1)}(q_k)\right]$ for $q_k \in q_{1:K}$.

end

The advantage of this approach, over NUNC Local, is that only K pieces of information are required to store information about the estimate of the CDF of the null distribution, irrespective of the number of points observed so far or the size of the sliding window, satisfying the memory constraint requirement of our application.

5.2.2 Parameter selection

The execution of both NUNC Local and Global require the selection of a various parameters, including the K quantiles $q_1, \ldots, q_k$ and threshold β. Additionally, the size of the sliding window W must be chosen with care. In practice, W be chosen

based on specific knowledge of the application and data generating process at hand. We defer further discussion of this until Section 5.3, where we consider the impact of W on different simulation scenarios.

Next we turn to the challenge of choosing the K quantiles $q_1, \ldots, q_k$. The value of K itself should be chosen to be proportionate to $\log(W)$, in line with the method proposed by Haynes et al. (2017a). In particular, the value $K = \lceil 4\log(n) \rceil$ was proposed, see Haynes et al. (2017a, Section 4.3) for details. Given K, one approach to choosing the $\{q_k\}$ would be to evaluate evenly spaced empirical quantiles. However, an alternative approach is motivated by Haynes et al. (2017a, Section 3.1)). That is, we select q_k such that

$$q_k = \hat{F}^{-1}\left(1 + (2W+1)\exp\left[\frac{c}{K}(2k-1)\right]\right)^{-1}, \tag{5.12}$$

where $c = -\log(2W-1)$. The reason for making such a choice is that this gives a higher weight to values in the tail of the distribution (Haynes et al., 2017a), allowing for more effective change detection. In the Local algorithm, the q_k will be updated as the window changes; in the Global algorithm, however, these K points are fixed in time. As such, the values of q_k must be obtained using the first W points of data the algorithm analyses. In some situations, however, this issue can be avoided because there is prior knowledge of the underlying distribution for the data. In this case known quantiles can be utilised rather than estimating them from the data.

Another important requirement for the two algorithms presented here, as in other sequential changepoint methods, is the ability to control the false alarm rate (Tartakovsky et al., 2014). In general, the value of β will be tuned so that the probability of a false alarm for data under the null hypothesis is set to some level α. This will be the case, for instance, in the telecommunications application where the threshold value will be tuned on devices where no even is detected. That said, we can follow a similar approach to that of Eichinger and Kirch (2018) to obtain an idea of how beta relates to the probability of a false alarm. Indeed, we can (asymptotically) approximate the distribution of each term in the sum of equation (5.6) by a chi-squared-1 distribution (Wilks, 1938). This is the asymptotic distribution of the likelihood-ratio test for a fixed t, τ, and q, assuming independent identically distributed (i.i.d.) data. With this approximation, it can be shown that: If β is chosen such that $\beta = \max\{\beta_1, \beta_2\}$, where

$$\beta_1 = 1 - 8K^{-1}\log\left(\frac{\alpha}{W(t-W+1)}\right)$$
$$\beta_2 = 1 + 2\sqrt{2\log\left(\frac{W(t-W+1)}{\alpha}\right)},$$

then the probability of a false detection by time t is bounded above by α.

Proof 2 *This result follows from bounds on the tail of sums of chi-squared distributions and a Bonferonni correction – see the Appendix for details.*

It should be noted, that in situations where the window size is large this bound may be conservative due to it being an asymptotic bound. Furthermore, when the assumptions of data independence and identical distribution are not met, this bound may not hold, as illustrated in simulations. In such settings we suggest selecting a threshold by tuning on a data stream that does not contain any changes.

If using an approximate grid of size $J < W$, then it is necessary to replace the values of W in the above proposition with the value J. Furthermore, as a corollary to Proposition 5.2.2, a bound can be obtained for use in NUNC Global.

Corollary 2 *If β is chosen such that $\beta = \max\{\beta_1, \beta_2\}$, where*

$$\beta_1 = 1 - 8K^{-1}\log\left(\frac{\alpha}{(t-W+1)}\right)$$
$$\beta_2 = 1 + 2\sqrt{2\log\left(\frac{(t-W+1)}{\alpha}\right)},$$

then the probability of a false detection by time t is bounded above by α.

Proof 3 *The proofs follows similarly to Proposition 5.2.2, however we perform only one test, rather than W tests, per window.*

Now that the methodology behind NUNC has been presented, and methods for quantile and threshold selection discussed, we consider its performance within various simulation settings.

5.3 Simulation Study

In this section we perform examine properties of both NUNC approaches in various simulation settings. These can be see in Figure 5.3. The first (Figure 5.3(a)) is a change in the mixture proportions of a bi-modal Gaussian distribution, whilst the second example considers a change in the scale of a Cauchy Distribution. The third setting is a change in the amplitude of a sinusoidal process, and the final setting is that of a change in the drift parameter of an Ornstein-Uhlenbeck (OU) process. Both the sinusoidal and OU examples are included to highlight how NUNC performs when the independence assumption is not met. Realisations of each of these data generating processes can be seen in Figure 5.3.

In what follows, we explore the performance of each method across the four given scenarios. In particular we consider the influence of window size on the power, and

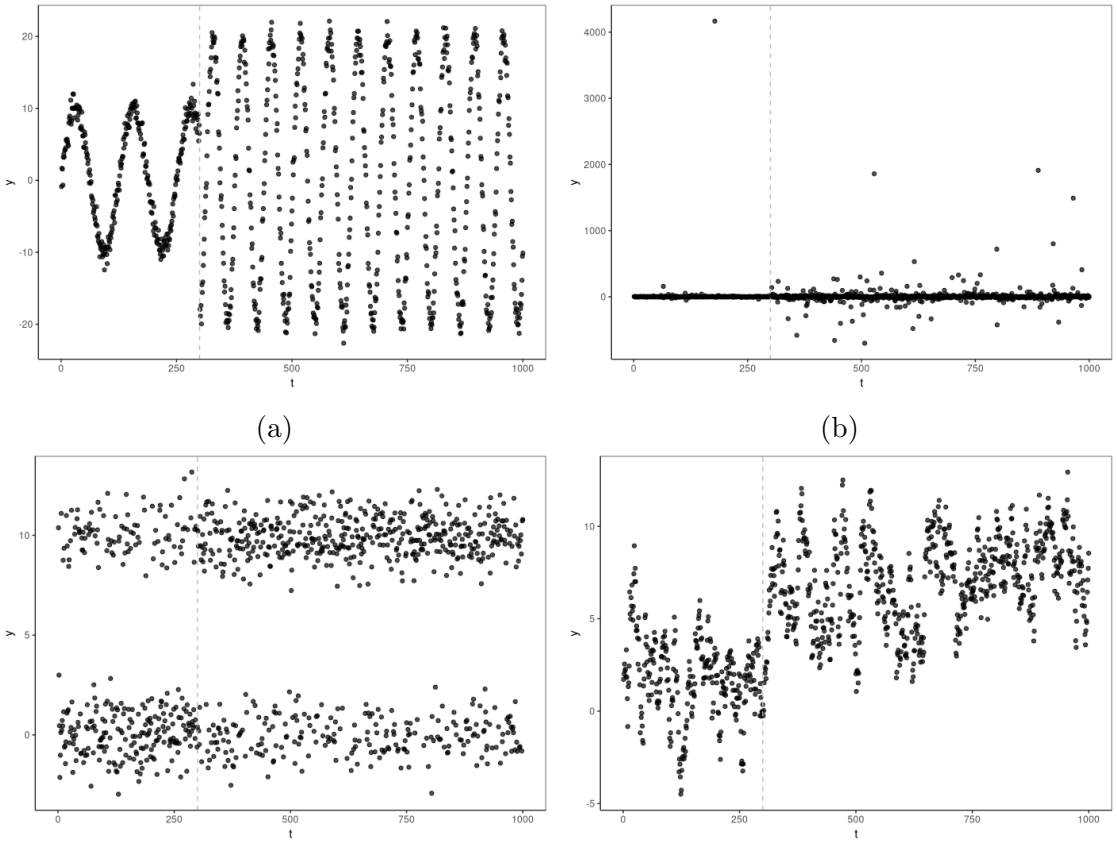


Figure 5.3: Four different simulations scenarios: (a) change in the amplitude of a sinusoidal process; (b) change in scale of a Cauchy distribution; (c) change in mixture proportions of a bi-modal Gaussian distribution; and (d) change in the drift parameter of an Ornstein–Uhlenbeck process.

the detection delay, of the test. In each setting we will also compare the NUNC-based tests against a MOSUM test, as implemented by Meier et al. (2021), a competitor nonparametric online changepoint algorithm that has a lightweight data footprint.

5.3.1 False Alarm Probability

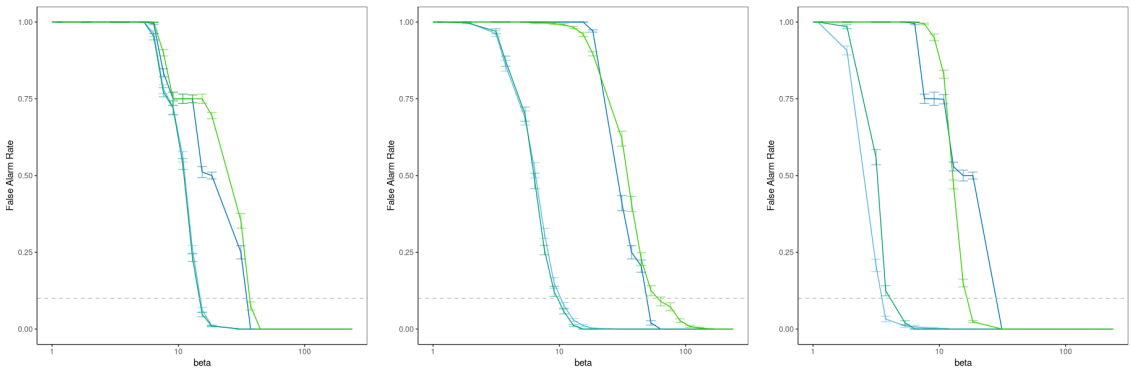
We begin by considering the false alarm rates returned by the three methods (NUNC Local, NUNC Global and MOSUM) for 100 replicates of each of our four data generating scenarios, without a change being present. In each case, the series generated was of length 1000, with $K = 20$ and $W = 150$ for NUNC Local and NUNC Global. For comparison, we also compared against the equivalent MOSUM procedure (i.e. $W = 150$, and other settings set to default). The resulting false alarm rates for a range of thresholds can be seen in Figure 5.4.

To explore the practical utility of Proposition 5.2.2, we compare the thresholds required for the i.i.d. multi-modal Gaussian and Cauchy change-in-scale false alarm rate when seeking to achieve a 10% false alarm rate. Our study highlights that penalty values of 9 and 10 are required by the NUNC Global algorithm for the multi-modal Gaussian and Cauchy scenarios respectively. In these two settings, penalties of 11.6 and 12.6 respectively were required for NUNC Local. This compares favourably with the approximate penalty values selected using Proposition 5.2.2 (9.51 and 12.30 for the Global and Local cases respectively). Unsurprisingly, in the case of the (non-i.i.d., temporally dependent) sinusoidal and OU scenarios, the penalties required for a 10% false alarm rate differ from those provided by Proposition 5.2.2.

5.3.2 Detection Power and Detection Delay

We now turn to consider the detection power and detection delay of the NUNC algorithms in a variety of settings. Following Tartakovsky et al. (2014), we define the detection power as the probability that a changepoint is detected after it has occurred, and the detection delay as the difference between the stopping time of the test and the time the changepoint is known to have emerged. Again we focus on 100 replicates of each of the four scenarios displayed in Figure 5.3, where each series is of length 1000 and the change occurs at time $t = 300$. In each case we seek to estimate the detection power and detection delay, controlling the false alarm rate at 10% and allowing the window size W of the algorithms to vary.

Results for the detection power, and detection delay, are summarised in Tables 5.1 and 5.2 respectively. It is notable that NUNC is able to detect changes in a variety of settings, including those where the data has time-dependent structure. We also note that NUNC Global outperforms NUNC Local in most cases, except when the underlying distribution is sinusoidal. This is perhaps to be expected since NUNC



(a) (b) (c)

Figure 5.4: The False Alarm rate for increasing threshold values for the four different simulation scenarios analysed with (a) NUNC Local, (b) NUNC Global and (c) MOSUM. The dotted line indicates a false alarm rate of 0.10, and the error bars indicate two standard deviations. In each plot the simulation scenarios are represented by sinusoidal change-in-amplitude (dark blue); the Cauchy change-in-scale (light blue); in emerald green, the change-in-mixture proportions (emerald green); and the change-in-drift in a OU process (light green).

Global incorporates the long-run empirical CDF which stores the historical data. This allows for better identification of departures from the null when the data is stationary. When the data is non-stationary, however, this is not so beneficial and so the performance of NUNC Local is comparable.

Turning to consider the results obtained for the detection delay, displayed in Table 5.2, it is evident that NUNC Local demonstrates stronger performance than that of NUNC Global. This is as expected, because NUNC Global checks if the distribution of the data in the window differs from the long run empirical CDF, whereas NUNC Local checks each point in the window (after pruning as per Proposition 1) for a changepoint within the window.

In comparison to the MOSUM, the detection power of NUNC typically exceeds it except in the specific case of multi-modal data being analysed with a large window. The reason MOSUM performs so well in this case is due to the fact that the change in mixture proportions can also be cast as a change in mean. For the sinusoidal process, however, the non-stationarity of the data means that the threshold that is required is too high for detection to take place.

The results in Table 5.1 also illustrate how, as one might expect, the performance of NUNC Local improves for stationary data as the size of the window increases. Specifically, the larger window provides a better estimate of the CDF of the (stationary) data stream, which in turn makes it easier to identify when a change

has occurred. The price for this increased power, however, comes in the form of an increase in computational cost due to the larger window size. As such, there is a trade off between detection power and the computational burden of NUNC Local. For NUNC Global, on the other hand, in many situations the use of the long run CDF provides a better estimate of the distribution under the null. This somewhat reduces the need to increase the window size.

5.4 Applications

5.4.1 Monitoring Operational Performances of Network Devices

We now revisit the telecommunications example, briefly introduced in Section 5.1, to explore the utility of NUNC in this setting. Recall that the data consists of historic records of a key operational metric routinely monitored on devices that have limited computational power and data storage capability. We have records for 473 such devices, of which 133 were known to contain a (series specific) event that triggered a user intervention. Due to the specifics of the application, engineers believed that it is possible to identify the start of the event in the operational data *before* a user identifies and makes an intervention. If this is true, then it would be desirable to identify the start of changing structure in advance of the user identifying and making an intervention. We call the correct detection of such a change in advance of user identification, an 'anticipation'. Conversely, the detection of such an event before it is resolved is called a 'detection'. The aim of this exploratory analysis, therefore, is to identify to what extent NUNC can (a) identify the correct (event-containing) series and (b) to what extent it can be used to 'anticipate' or 'detect'.

Before summarising the results, we briefly discuss the various parameter choices made: specifically, the threshold β, the window size W, and the choice of K. In line with Haynes et al. (2017a), we choose $K = \lceil 4 \log(W) \rceil$. The choice of β was made to control the false alarm rate at a desired level after discussion with domain experts. In this particular setting, false alarms can be tolerated if this results in improved identification of real events. Consequently a false alarm rate of 15% was selected. In order to identify the appropriate value of β to achieve this, we first fix a window size and then perform NUNC on the 340 data series without the event, choosing a value of β that gives the desired false alarm rate. NUNC is then applied to the 133 series known to contain an event using this β, for the chosen window size, to explore the power of the approach for different window sizes. A similar process is also used to implement the MOSUM test; again, the threshold is chosen to control the false alarm rate at 15% for a given window size.

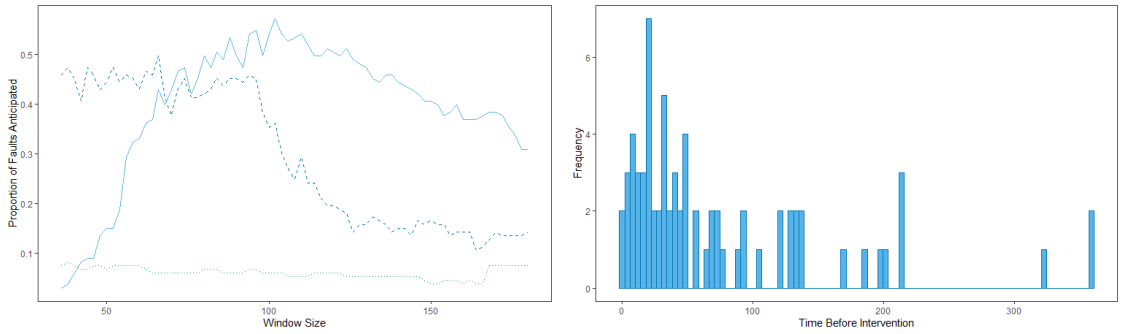
In Figure 5.5(a) we present the anticipation rate for a range of window sizes. As

Window	50			100			150			200		
Process	Local	Global	MOSUM	Local	Global	MOSUM	Local	Global	MOSUM	Local	Global	MOSUM
Cauchy	0.56	**0.9**	0.02	0.71	**0.89**	0.05	0.81	**0.91**	0.02	0.85	**0.91**	0.01
Multimodal	0.45	**0.87**	0.43	0.31	**0.79**	0.71	0.58	0.8	**0.92**	0.58	0.83	**0.92**
Sinusoidal	**0.95**	0	0.93	**1**	0.92	0	0.17	**0.91**	0	**0.98**	0.92	0
OU	0.25	**0.92**	0.42	0.47	**0.91**	0.68	0.36	**0.9**	0.86	0.79	**0.93**	**0.93**

Table 5.1: Detection power of NUNC Local, NUNC Global, and the MOSUM for various window sizes, with the best performance for each scenario highlighted in bold.

Window	50			100			150			200		
Process	Local	Global	MOSUM	Local	Global	MOSUM	Local	Global	MOSUM	Local	Global	MOSUM
Cauchy	150.16	**67.13**	345	**51.67**	109.39	211	**72.99**	133.64	494	**48.96**	143.69	285
Multimodal	285.29	**131.46**	169.21	285.82	238.63	**49.23**	242.59	266.96	**48.03**	206.09	297.05	**59.07**
Sinusoidal	**19.59**	Inf	411.69	**5**	102.03	Inf	**9.16**	159.3	Inf	**7.58**	238.61	Inf
OU	284.32	**115.88**	173.88	246.71	136.81	**77.65**	242.85	173.01	**78.83**	155.7	221.06	**81.34**

Table 5.2: Average detection delay of NUNC Local, NUNC Global, and the MOSUM for various window sizes, with the best performance for each scenario highlighted in bold. Note that in line with Tartakovsky et al. (2014), if no detection takes place then this corresponds to a detection delay of infinity.



(a) (b)

Figure 5.5: In (a) comparison of anticipation rates achieved by the Local (Solid Line) and Global (Dashed Line) variants of NUNC, and the MOSUM (Dotted Line), for varying window sizes, for a window size of 100 and a false alarm rate of 15%. In (b) a histogram illustrating the distribution of the time between the detection of an event by NUNC Local and the report by a customer, for a window size of 100 and a false alarm rate of 15%.

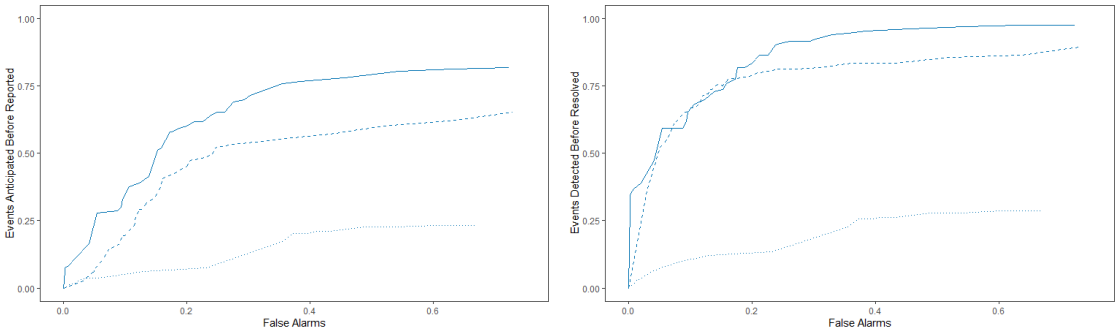
can be seen, a window of size between 80 and 120 performs the best, with NUNC Local correctly identifying (i.e. anticipating) $> 50\%$ of events in advance of user intervention. We also note that NUNC outperforms the MOSUM for various choices of W. One reason for this is because the MOSUM test threshold is set to avoid detecting the non-anomalous changes in mean that many of the series exhibit, and this reduces detection power.

False Alarms	1%	5%	10%	15%
Local	0.08 (0.37)	0.28 (0.59)	0.38 (0.68)	0.51 (0.77)
Global	0.03 (0.36)	0.06 (0.51)	0.20 (0.67)	0.35 (0.76)
MOSUM	0.02 (0.02)	0.04 (0.08)	0.05 (0.10)	0.06 (0.12)

Table 5.3: Table illustrating proportion of events anticipated (and detected) for varying rates of false alarms for both NUNC Local and NUNC Global.

Finally, for a fixed window size ($W = 100$) we explore the anticipation and detection rate as the false alarm rate (or equivalently β) varies. The results are summarised in Table 5.3, and Figure 5.6.

From the results presented, one remark that can be made is that the detection power for NUNC Local and NUNC Global is similar, with both achieving over 75% for a false alarm rate of 15% and window of $W = 100$, but the anticipation power of NUNC Local is significantly better for a range of false alarm rates and window sizes. As in the simulations on detection delay, this is due to the way that NUNC Local



(a) (b)

Figure 5.6: Comparison of event anticipation (a) and event detection (b) rates achieved by both the Local (Solid Line) and Global (Dashed Line) variants of NUNC, and the MOSUM (dotted line), for a window of size $W = 100$ and a false alarm rate of 15%.

checks every point within the window for a change, allowing for a shorter detection delay. A second observation is that NUNC achieves better results than the MOSUM in terms of both anticipation, and power, as the false alarm rate varies.

5.4.2 Controller Data Analysis

In this section we perform an analysis of a DualShock controller movement dataset. As described in the introduction, the corpus contains 100 data streams consists of 2000 observations each. The aim of the analysis is to detect a change in the physical state of the controller, which could be either directly or indirectly associated with the presence of a user.

To measure the performance of NUNC we perform NUNC Local, the NUNC local approximation from 5.7 (referred as NUNC Approx), NUNC Global, and the MOSUM on each data stream. We record a successful detection if a change is identified after the known time of the change, we record the detection delay as the difference between the stopping time and the time at which the real event occurred. In Table 5.4 we present the results for the detection test using $K = 15$, a window of size 100, for NUNC Approx $J = 5$, and the β value set according to Proposition to give a false alarm rate of 1%. To allow for a comparison with the MOSUM procedure we tune its threshold so that the false alarm rate is controlled at 1% on the data streams before movement occurs.

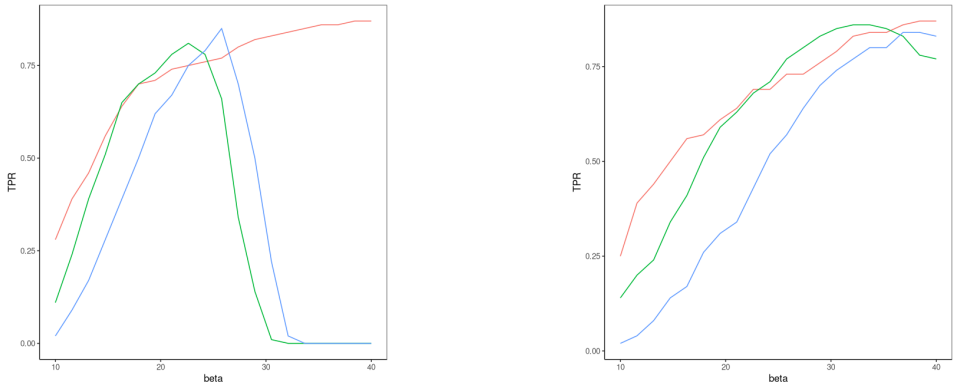
As can be seen from the results in Table 5.4, the performance of NUNC outstrips that of the competitor. Furthermore, NUNC Local is able to offer the best performance in terms of power but it is only marginally better than NUNC Global.

Method	Power	Delay	Delay sd
NUNC Local	0.79	98.68	162.63
NUNC Approx	0.77	149.35	221.45
NUNC Global	0.76	90.71	100.76
MOSUM	0.31	96.68	179.67

Table 5.4: Table depicting the performance of the variants of NUNC and the MOSUM on the controller movement dataset.

We also see that NUNC Approx offers comparable detection power with NUNC Local, for a saving of as many as 95×1901 checks for a change, however the use of a smaller grid drastically increases the detection delay.

One aspect of NUNC Local that can also be explored in this application is how the detection power can be affected by both too small, or too large, a penalty value. Indeed, too small a penalty results in early detections that are recorded as false positives and reduce detection power, whereas too large a penalty results in missed detections. This is depicted in Figure 5.7. The problem is less apparent for the Global variant, however, because of the use of the historic information stored in the long run CDF. Finally, we observe that the detection power for larger thresholds is increased for NUNC Local for a larger window; this is because a larger window contains the changepoint for a longer period of time, offering more opportunities for detection at the expense of enhanced computational cost. By fixing the number of checks completed, however, NUNC Approx affords this extra power without this extra cost. This demonstrates the power of the method.



(a) (b)

Figure 5.7: Detection power of NUNC Local (blue), NUNC Global (red), and NUNC Approx (green) for a window size of 100 (a) and 200 (b) with respect to changing values of β.

Chapter 6

Remarks and Conclusions

Three novel methodologies were introduced during the course of this thesis: (a) DeCAFS, extending the FPOP recursion to a more sophisticated model accounting for both autocorrelation in the noise and a fluctuating mean signal; (b) FOCuS, extending the sequential Page-CUSUM statistics to an online analysis to deal with a pre-change mean unknown and to more sophisticated models; and (c) NUNC Local and Global: two related, nonparametric changepoint methods with a lightweight data footprint.

Concerning DeCAFS and FOCuS, there are various ways of developing the recursions, that build on other extensions of the functional pruning version of optimal partitioning. For example, conditions on the underlying mean object of inference could be implemented to produce inference constrained to specific change patterns (Hocking et al., 2020), or a geometric decay on the mean between segments (Jewell et al., 2020). Of major interest is a derivation of a multidimensional recursion, as this is one of the limiting factors of the functional pruning recursions, as they are currently restricted to the one-dimensional case. This could be solved either exactly thanks to the recent findings of Runge (2020), or through a grid-based approximation on the domain of the parameters.

Ultimately, based on the latest functional pruning developments, the goal is to derive a versatile offline and online multivariate changepoint detection method. Ideally, such a method should be capable of dealing with non-standard change patterns and scenarios, various stochastic processes, whilst also being robust to outliers and model misspecification. One possible way of doing so would be to embed the online changepoint problem within a discrete-state hidden Markov model as seen in Runge et al. (2020a), encoding all the novel recursions on the different state transitions.

Concerning NUNC, we denote how the Global implementation offers greater power in instances where there is a stationary underlying null distribution for the data (*cf.* Section 5.3.2). Conversely NUNC Local shows greater resilience and is able to outperform NUNC Global in settings where the process contains time-dependent

or other non-independent structures, such as in the examples that we considered. As with any approach, NUNC has various weaknesses that might be identified for criticism. For example, the estimation of the empirical CDF from windowed data means that gradual changes are likely to go undetected. In addition, as identified by our simulation study, NUNC Global struggles with changing structure in time-dependent series. The investigation of potential alternatives to the NUNC framework, that can resolve such weaknesses, are left as avenues for future research.

Lastly, some work could be done in order to provide better initialization values for each method, which could lead to further improvements on the statistical performance of each method respectively. For instance, work could be done to improve the initial estimation of parameters σ_η, σ_ν and ϕ for DeCAFS, or providing a procedure to pick W the window size and K the number of quantiles for NUNC. This is particularly challenging in an online scenario, as no look-up-ahead is allowed and therefore there is the need for online estimators to learn and adapt parameters iteratively.

Appendix A

DeCAFS

A.1 Proof of Proposition 1

The initial condition for $Q_1(\mu)$ follows immediately from its definition.

Then, for $t \in \{2, ..., n\}$, we need to condition the problem separately on whether or not we have a changepoint. If we consider no change in the mean of the signal, then we can we can re-arrange the cost at time t based on the cost at time $t-1$ in the following way:

$$Q_t(\mu|\delta_t = 0) = \min_u \left\{ Q_{t-1}(u) + \lambda(\mu - u)^2 + \gamma\Big((y_t - \mu) - \phi(y_{t-1} - u)\Big)^2 \right\} .$$

Similarly, when we have a change:

$$\begin{aligned} Q_t(\mu|\delta_t \neq 0) &= \min_{u,\delta} \left\{ Q_{t-1}(u) + \lambda(\mu - u - \delta)^2 + \gamma\Big((y_t - \mu) - \phi(y_{t-1} - u)\Big)^2 + \beta \right\} \\ &= \min_u \left\{ Q_{t-1}(u) + \gamma\Big((y_t - \mu) - \phi(y_{t-1} - u)\Big)^2 + \beta \right\} \end{aligned}$$

where the second equality comes from minimising over δ.

Lastly, to obtain the whole cost at time t we take the minimum of these two functions:

$$\begin{aligned} Q_t(\mu) &= \min \{Q_t(\mu|\delta_t = 0), Q_t(\mu|\delta_t \neq 0)\} \\ &= \min_u \left\{ Q_{t-1}(u) + \min\{\lambda(\mu - u)^2, \beta\} + \gamma\Big((y_t - \mu) - \phi(y_{t-1} - u)\Big)^2 \right\} . \end{aligned}$$

□

A.2 Proof of Proposition 2

From the result obtained in Appendix A.1, simple, albeit tedious, algebraic manipulation enables us to re-write the recursions for $Q_t(\mu|\delta_t \neq 0)$ and $Q_t(\mu|\delta_t = 0)$ in terms of the infimal convolution operator. Let $z_t = y_t - \phi y_{t-1}$.

For $Q_t(\mu|\delta_t \neq 0)$, we can rearrange

$$\begin{aligned}
&\gamma\Big((y_t - \mu) - \phi(y_{t-1} - u)\Big)^2 = \gamma(z_t - \mu + \phi u)^2 \\
&= \gamma(z_t - \mu)^2 + \gamma\phi^2 u^2 + 2\gamma\phi u z_t - 2\gamma\phi u\mu \\
&= \gamma(z_t - \mu)^2 + \gamma\phi^2 u^2 + 2\gamma\phi u z_t + \gamma\phi(u - \mu)^2 - \gamma\phi u^2 - \gamma\phi\mu^2 \\
&= \gamma\phi(u - \mu)^2 - \gamma\phi(1 - \phi)\left(u - \frac{z_t}{1 - \phi}\right)^2 + \gamma\phi\frac{z_t^2}{1 - \phi} + \gamma(z_t - \mu)^2 - \gamma\phi\mu^2
\end{aligned}$$

Hence, we have

$$\begin{aligned}
Q_t(\mu|\delta_t \neq 0) &= \min_{u\in\mathbb{R}}\left[Q_{t-1}(u) - \gamma\phi(1 - \phi)\left(u - \frac{z_t}{1 - \phi}\right)^2 + \gamma\phi(u - \mu)^2\right] \\
&\quad + \frac{\gamma}{1 - \phi}(z_t - (1 - \phi)\mu)^2 + \beta \\
&= \mathsf{INF}_{\mathbb{Q}_{t-1},\gamma\phi}(\mu) + \frac{\gamma}{1 - \phi}\Big(z_t - (1 - \phi)\mu\Big)^2 + \beta = Q_t^{\neq}(\mu),
\end{aligned}$$

where

$$\mathbb{Q}_{t-1}(u) = Q_{t-1}(u) - \gamma\phi(1 - \phi)\left(u - \frac{z_t}{1 - \phi}\right)^2.$$

Similar, for $Q_t(\mu|\delta_t = 0)$, we can rearrange

$$\begin{aligned}
&\lambda(\mu - u)^2 + \gamma\Big((y_t - \mu) - \phi(y_{t-1} - u)\Big)^2 \\
&= (\gamma\phi + \lambda)(u - \mu)^2 - \gamma\phi(1 - \phi)\left(u - \frac{z_t}{1 - \phi}\right)^2 + \gamma\phi\frac{z_t^2}{1 - \phi} + \gamma(z_t - \mu)^2 - \gamma\phi\mu^2.
\end{aligned}$$

Hence

$$Q_t(\mu|\delta_t = 0) = \mathsf{INF}_{\mathbb{Q}_{t-1},\gamma\phi+\lambda}(\mu) + \frac{\gamma}{1 - \phi}\Big(z_t - (1 - \phi)\mu\Big)^2 = Q_t^{=}(\mu),$$

where $\mathbb{Q}_{t-1}$ is defined above. □

Comment 1 *If $\phi < 0$ then $\gamma\phi < 0$ and the infimal convolution $\mathsf{INF}_{\mathbb{Q}_{t-1},\gamma\phi}(\mu)$ is not defined. In this case we make a transformation of variable $v = -u$ so that*

$$\begin{aligned}
Q_t(\mu|\delta_t \neq 0) &= \min_{v\in\mathbb{R}}\left[Q_{t-1}(-v) - \gamma\phi(1-\phi)\left(-v - \frac{z_t}{1-\phi}\right)^2 + \gamma\phi(-v-\mu)^2\right] \\
&\quad + \frac{\gamma}{1-\phi}(z_t - (1-\phi)\mu)^2 + \beta \\
&= \min_{v\in\mathbb{R}}\left[Q_{t-1}(-v) - \gamma\phi(1-\phi)\left(v + \frac{z_t}{1-\phi}\right)^2 - |\gamma\phi|(v^2 + 2\mu v + \mu^2)\right] + \frac{\gamma}{1-\phi}(z_t - (1-\phi)\mu)^2 + \\
&= \min_{v\in\mathbb{R}}\left[Q_{t-1}(-v) - \gamma\phi(1-\phi)\left(v + \frac{z_t}{1-\phi}\right)^2 + |\gamma\phi|(v^2 - 2\mu v + \mu)^2 + 2\gamma\phi(v^2+\mu^2)\right] \\
&\quad + \frac{\gamma}{1-\phi}(z_t - (1-\phi)\mu)^2 + \beta \\
&= \min_{v\in\mathbb{R}}\left[Q_{t-1}(-v) - \gamma\phi(1-\phi)\left(v + \frac{z_t}{1-\phi}\right)^2 + 2\gamma\phi v^2 + |\gamma\phi|(v-\mu)^2\right] \\
&\quad + \frac{\gamma}{1-\phi}(z_t - (1-\phi)\mu)^2 + 2\gamma\phi\mu^2 + \beta \\
&= \mathsf{INF}_{\tilde{\mathbb{Q}}_{t-1},|\gamma\phi|}(\mu) + \frac{\gamma}{1-\phi}\Big(z_t - (1-\phi)\mu\Big)^2 + 2\gamma\phi\mu^2 + \beta,
\end{aligned}$$

where

$$\tilde{\mathbb{Q}}_{t-1}(u) = Q_{t-1}(-u) - \gamma\phi(1-\phi)\left(u + \frac{z_t}{1-\phi}\right)^2 + 2\gamma\phi u^2.$$

Similarly if $\gamma\phi + \lambda < 0$ then we need to make a similar change to the equation for $Q_t(\mu|\delta_t = 0)$. This becomes

$$Q_t(\mu|\delta_t = 0) = \mathsf{INF}_{\bar{\mathbb{Q}}_{t-1},|\gamma\phi+\lambda|}(\mu) + \frac{\gamma}{1-\phi}\Big(z_t - (1-\phi)\mu\Big)^2 + 2(\gamma\phi+\lambda)\mu^2,$$

where

$$\bar{\mathbb{Q}}_{t-1}(u) = Q_{t-1}(-u) - \gamma\phi(1-\phi)\left(u + \frac{z_t}{1-\phi}\right)^2 + 2(\gamma\phi+\lambda)v^2.$$

If $\phi < 0$ then $\gamma\phi < 0$ and the infimal convolution $\mathsf{INF}_{\mathbb{Q}_{t-1},\gamma\phi}(\mu)$ is not defined. In this case we make a transformation of variable $\tilde{u} = -u$ and $\tilde{\phi} = -\phi$ so that

$$\begin{aligned}
&\gamma\Big((y_t - \mu) - \phi(y_{t-1} - u)\Big)^2 = \gamma(z_t - \mu + \phi u)^2 = \gamma(z_t - \mu + \tilde{\phi}\tilde{u})^2 \\
&= \gamma\phi(\tilde{u} - \mu)^2 - \gamma\tilde{\phi}(1-\tilde{\phi})\left(\tilde{u} - \frac{z_t}{1-\tilde{\phi}}\right)^2 + \gamma\tilde{\phi}\frac{z_t^2}{1-\tilde{\phi}} + \gamma(z_t-\mu)^2 - \gamma\tilde{\phi}\mu^2,
\end{aligned}$$

by the same manipulation as given at the start of this section.

Thus using that $Q_{t-1}(u) = Q_{t-1}(\tilde{u})$ we obtain

$$\begin{aligned} Q_t(\mu|\delta_t \neq 0) &= \min_{\tilde{u}\in\mathbb{R}} \left[Q_{t-1}(-\tilde{u}) - \gamma\tilde{\phi}(1-\tilde{\phi}) \left(\tilde{u} - \frac{z_t}{1-\tilde{\phi}} \right)^2 + \gamma\tilde{\phi}(\tilde{u}-\mu)^2 \right] \\ &\quad + \frac{\gamma}{1-\tilde{\phi}}(z_t - (1-\tilde{\phi})\mu)^2 + \beta \\ &= \mathsf{INF}_{\tilde{\mathbb{Q}}_{t-1},\gamma\tilde{\phi}}(\mu) + \frac{\gamma}{1-\tilde{\phi}}\Big(z_t - (1-\tilde{\phi})\mu\Big)^2 + \beta = Q_t^{\neq}(\mu), \end{aligned}$$

where

$$\tilde{\mathbb{Q}}_{t-1}(u) = Q_{t-1}(-u) - \gamma\tilde{\phi}(1-\tilde{\phi}) \left(\tilde{u} - \frac{z_t}{1-\tilde{\phi}} \right)^2.$$

Similarly, using the same transformation $\tilde{u} = -u$ and $\tilde{\phi} = -\phi$ we also derive

$$Q_t(\mu|\delta_t = 0) = \mathsf{INF}_{\tilde{\mathbb{Q}}_{t-1},\gamma\tilde{\phi}+\lambda}(\mu) + \frac{\gamma}{1-\tilde{\phi}}\Big(z_t - (1-\tilde{\phi})\mu\Big)^2 = Q_t^{=}(\mu),$$

where $\tilde{\mathbb{Q}}_{t-1}$ is defined above. □

A.3 Algorithm for $\mathsf{INF}_{\mathsf{Q}_{t,\omega}}$

Algorithm 11 shows how we can now calculate $\mathsf{INF}_{\mathsf{Q}_{t,\omega}}$ in a linear-in-piece $O(s)$ time complexity. In this algorithm we have input $q_*^i = \mathsf{INF}q_t^i$, where q_t^i is the i^{th} piecewise quadratic from Q_t with $i \in \{1, ..., s\}$. Algorithm 11 computes the intervals, DOM_*^i such that $\{\text{DOM}_*^{u_i}, i = 1, ..., s^*\}$ is the partition of the real line for $\mathsf{INF}_{\mathsf{Q}_{t,\omega}}$, with Q_* storing the associated quadratics for each interval in this partition. In Algorithm 11 we use the list-operator $Last(l)$ to designate the last element of the list l; $index\,Last(l)$,

delete Last(l) to get the associated index of the last element or to delete this element.

Algorithm 11: $\mathsf{INF}_{\mathsf{Q}_{t,\omega}}$ pruning

Input: List of ordered quadratics $(q_*^1, q_*^2, \ldots, q_*^{s-1}, q_*^s)$
begin Initialization: Q_* means "Remaining quadratics" and LB "Left Bound"
  $Q_* \leftarrow (q_*^1)$; $LB \leftarrow (-\infty)$
end
for $i = 2$ *to* s **do**
  $j \leftarrow index\, Last(Q_*)$
  $\mu_i : q_*^i(\mu_i) - q_*^j(\mu_i) = 0$ with $q_*^i(\mu) < q_*^j(\mu)$ for $\mu > \mu_i$ close to μ_i
  while $\mu_i <$ *Last(LB)* **do**
    delete Last(Q_*); *delete Last*(LB)
    $j \leftarrow index\, Last(Q_*)$
    $\mu_i : q_*^i(\mu_i) - q_*^j(\mu_i) = 0$ with $q_*^i(\mu) < q_*^j(\mu)$ for $\mu > \mu_i$ close to μ_i
  end
  $Q_* \leftarrow (Q_*, q_*^i)$; $LB \leftarrow (LB, \mu_i)$
end
$s^* = \#LB$ (the number of element in LB)
for $i = 1$ *to* $s^* - 1$ **do**
  $\text{DOM}_*^i =]LB(i), LB(i+1)]$
end
$\text{DOM}_*^{s^*} =]LB(s^*), +\infty[$
Return Q_* and $(\text{DOM}_*^1, ..., \text{DOM}_*^{s^*})$

A.4 Additional Empirical Results

A.4.1 Distorted Parameter Estimation

To see what might happen in case of a distorted parameter estimation, as mentioned in the simulation study of Section 3.6, please refer to Figure A.1. We can see there, how even when misspecifying the model, in this case via fitting a pure AR(1) when there was some drift in the signal, we find a distorted signal μ estimation, however we are still able to reconstruct the changepoint locations relatively well. A complete simulation study on the behaviour of the estimator is in the supplementary materials of Romano et al. (2021).

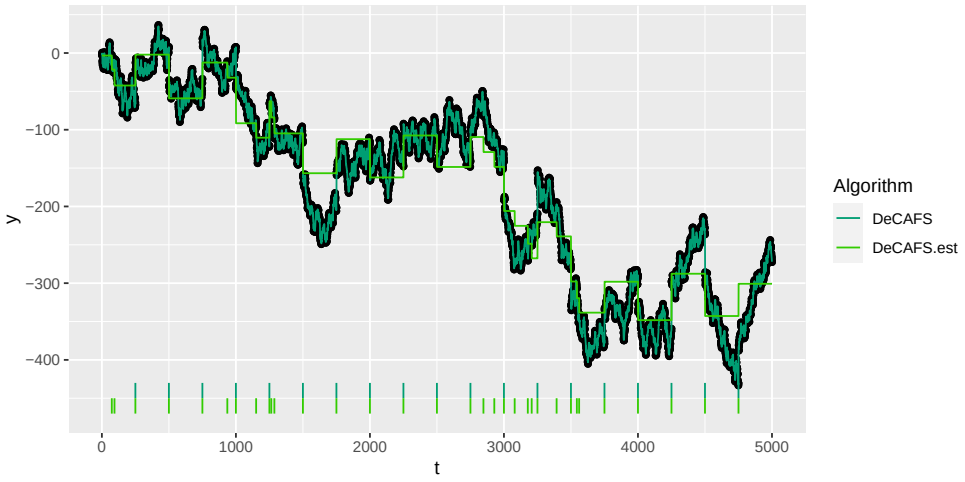


Figure A.1: An example of a sequence generated with $\sigma_\eta = 4,\ \sigma_\nu = 2,\ \phi = 0.14$, with relative signal and changepoints estimates of DeCAFS with real parameter values compared to DeCAFS with estimated ones. On this particular sequence, our estimator returns values for initial parameters of $\hat{\sigma}_\eta = 0,\ \hat{\sigma}_\nu = 4.6,\ \hat{\phi} = 0.98$, resulting in a distorted signal estimation.

A.4.2 Comparison of DeCAFS and AR1Seg on a Ornstein-Uhlenbeck process

We compare performances of both DeCAFS and AR1Seg from Chakar et al. (2017) on a discrete Ornstein-Uhlenbeck process with abrupt changes. Let $y_{1:n} = (y_1, \ldots, y_n) \in \mathbb{R}^n$ a sequence of n realizations of the process:

$$y_t = \mu_t + \epsilon_t \quad t = 1, \ldots, n$$

where for $t = 2, \ldots, n$

$$\mu_t = f_t + \nu_t$$

and $\epsilon_t \underset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$, f_t is a piecewise constant signal we wish to infer the changes of whether $f_t \neq f_{t-1}$, and finally ν_t is a discrete Ornstein–Uhlenbeck process defined by:

$$\nu_t = \nu_{t-1} - \theta\nu_{t-1} + \sigma_\nu \eta_t; \quad \text{with } \eta_t \underset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\eta^2).$$

Differently from the RW process introduced in the main model in Equation 3.1 the OU process is a mean reverting process, which rather then diverging as a pure Random Process would do, it reverts to its original initial value. This is regulated by

the parameter θ, where it can be seen that for $\theta = 0$ we observe a pure Random Walk process.

We performed a small simulation study comparable to the previous ones, which is summarised in Figure A.2, where we report the average F1 scores of DeCAFS and AR1Seg over 100 replicates of each experiment. Separate figures for precision and recall can be found in Appendix A.5, Figure A.7.

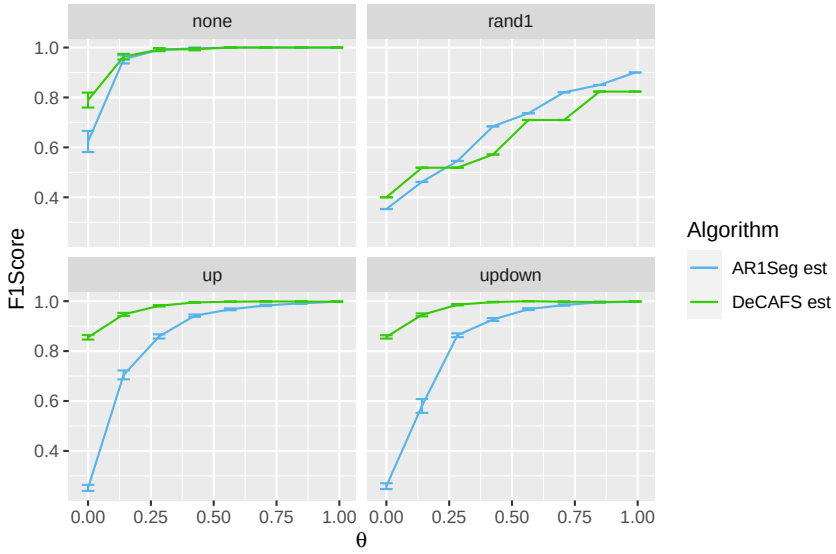


Figure A.2: F1 score on different scenarios with an underlying OU process as we vary θ. Data simulated fixing $\sigma_\nu = 1$, $\sigma_\eta = 1$ and $\sigma = 1$ over a change of size 10.

We denote how DeCAFS is relatively robust to this kind of model misspecification, producing good changepoints estimates overall, especially for larger values of θ. As a matter of fact, for $\theta \approx 1$ we have in fact a simple AR(1) noise with changes: in this scenario AR1Seg matches DeCAFS performances.

A.5 Additional Simulation Results

In Figures A.3 and A.4 we summarize the results of the first simulation of Section 3.6 in terms of Precision (the proportion of detected changes which are correct) and Recall (the proportion of true changes that are detected). Similarly, Figure A.5 shows Precision and Recall for the simulation with an AR(2) noise, Figure A.6 shows Precision and Recall for the simulation with an underlying sinusoidal signal, and Figure A.7 shows Precision and Recall for the simulations where the local fluctations in the mean are from an Ornstein-Uhlenbeck process.

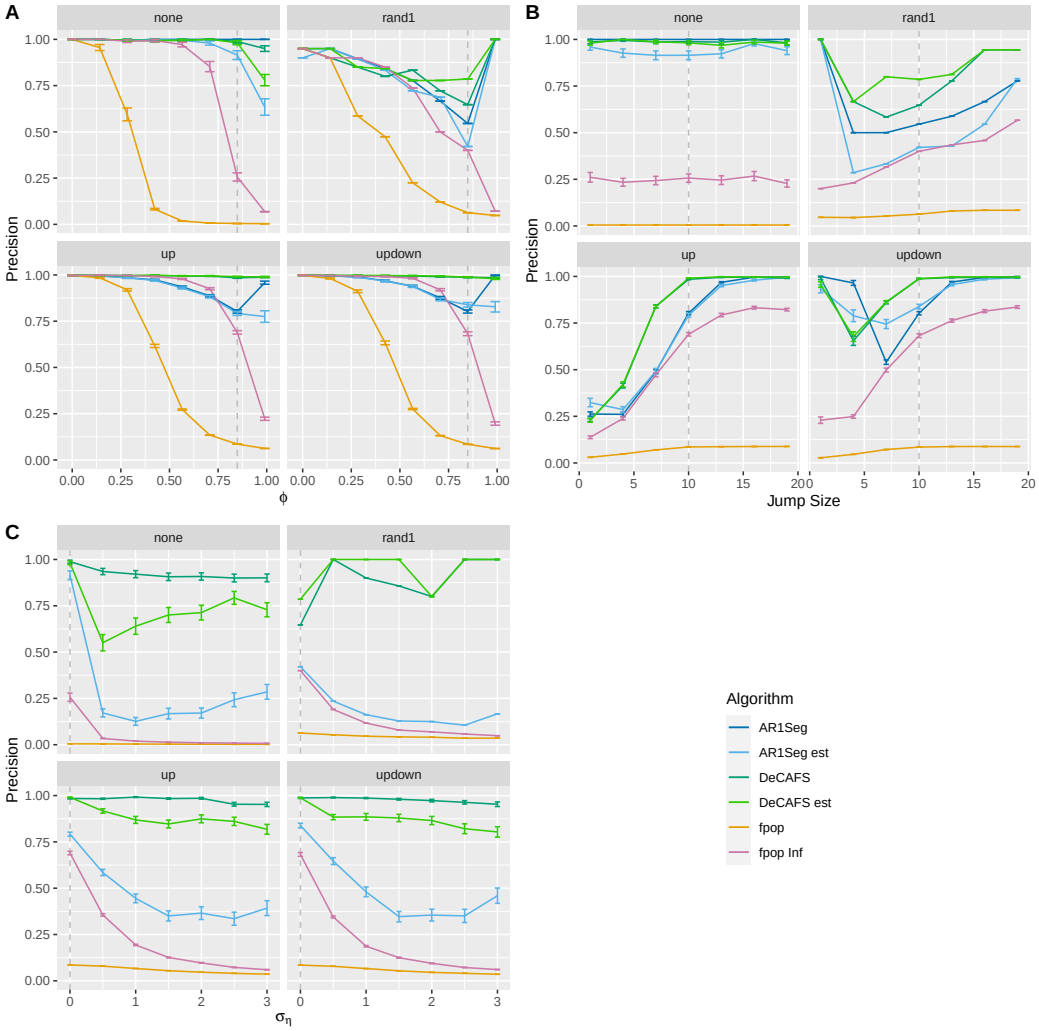


Figure A.3: Precision on the 4 different scenarios from the main simulation study of Section 3.6. Should be read in conjunction with Figure 3.4.

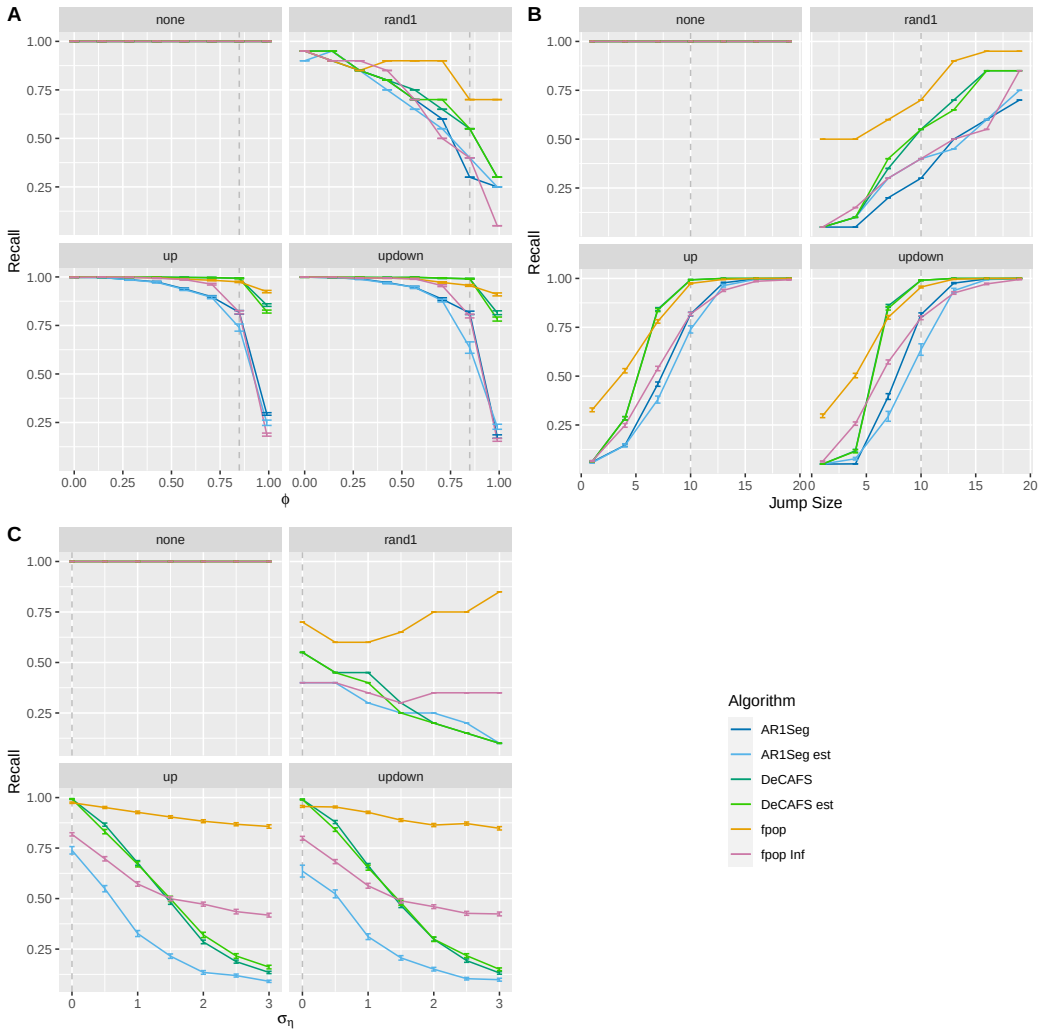
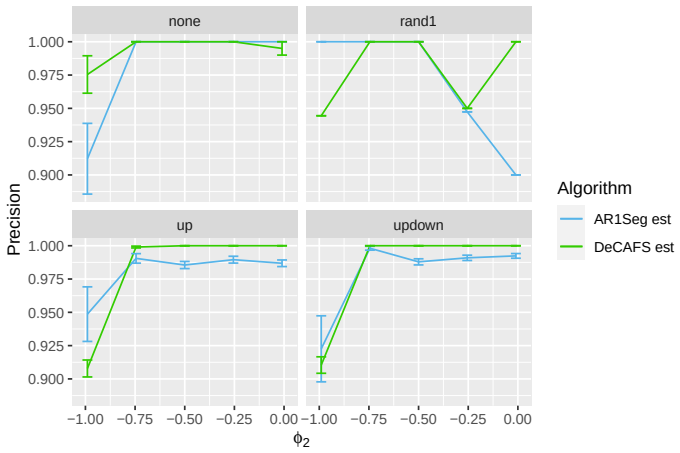
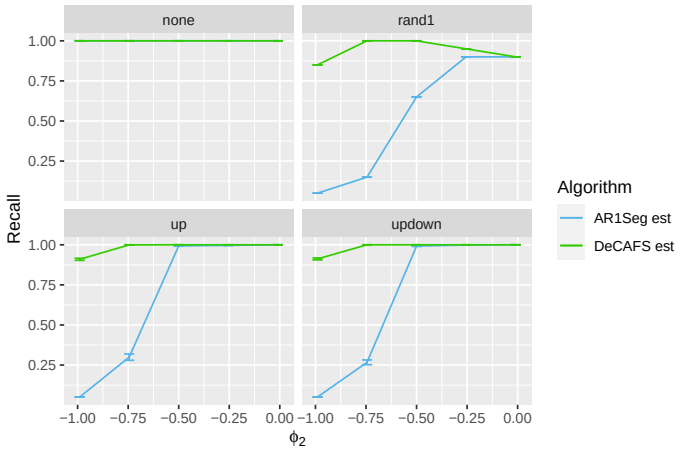


Figure A.4: Recall on the 4 different scenarios from the main simulation study of Section 3.6. Should be read in conjunction with Figure 3.4.

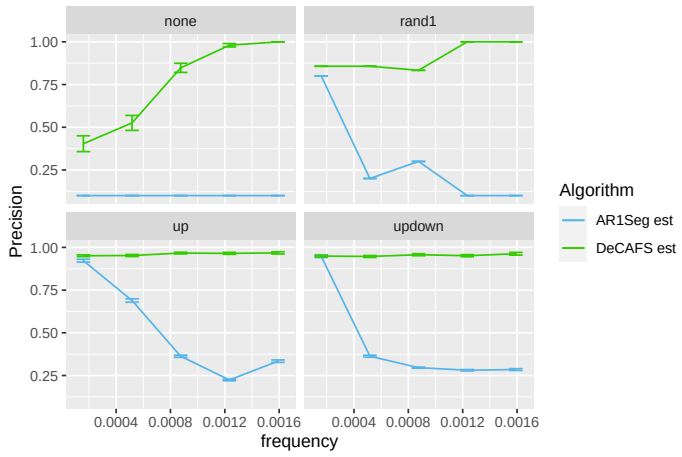


(a)

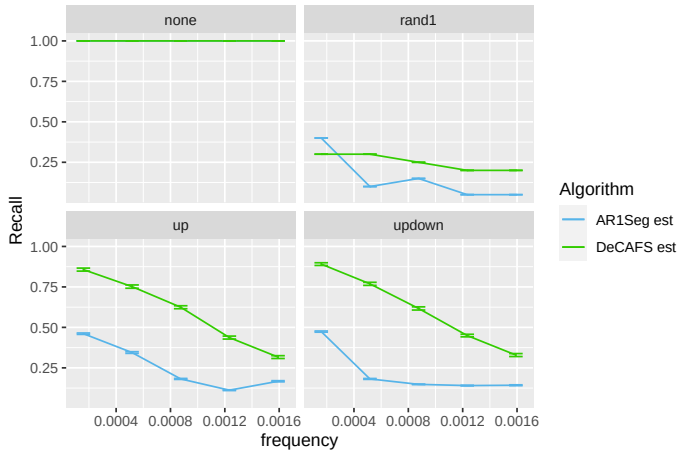


(b)

Figure A.5: Precision (a) and Recall (b) on different scenarios with a AR(2) noise. Should be read in conjunction with Figure 3.5.

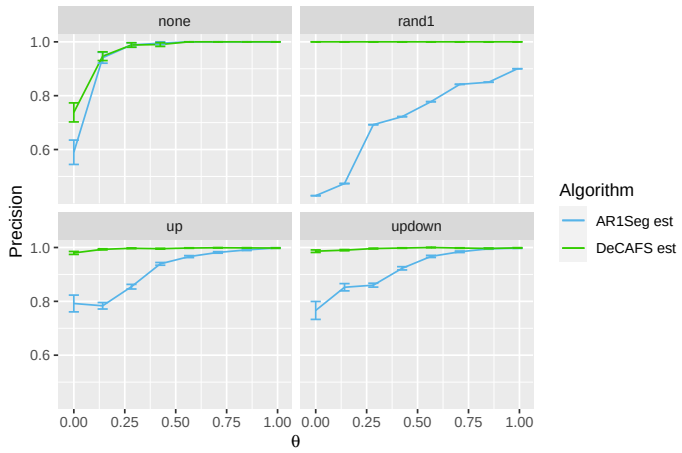


(a)

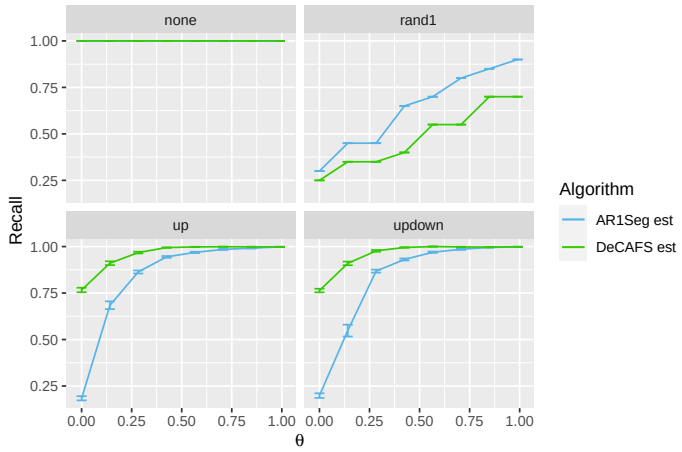


(b)

Figure A.6: Precision (a) and Recall (b) on different scenarios with an underlying sinusoidal process. Should be read in conjunction with Figure 3.6.



(a)



(b)

Figure A.7: Precision (a) and Recall (b) on different scenarios with an underlying Ornstein-Uhlenbeck process. Should be read in conjunction with Figure A.2.

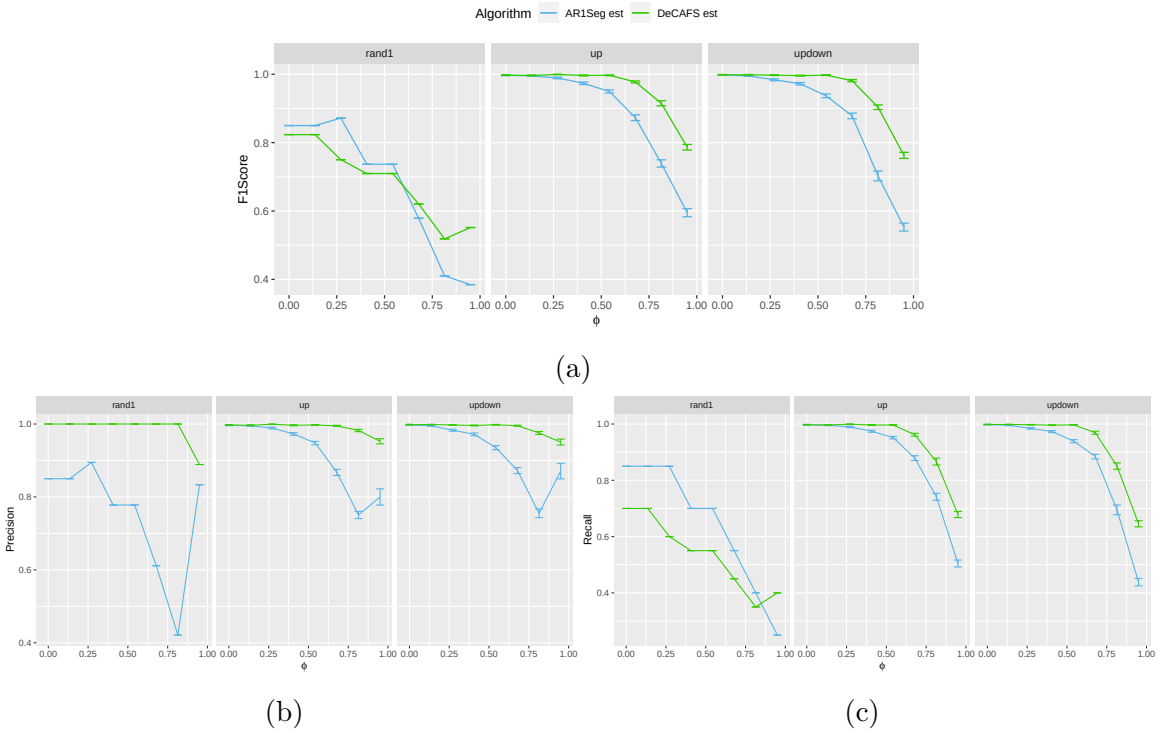


Figure A.8: F1 score (a), Precision (b) and Recall (c) on 3 different change scenarios with an independent between-the-changes AR(1) noise as we vary ϕ. Data simulated fixing $\sigma_\nu = 2$ over a change of size 10.

As an extension on the simple AR(1) noise (Figure 3.4 A), we investigate a further case of model misspecification. Differently to what already shown, we now assume independence in the AR(1) noise across the various segments. Results for F1Score, Precision and Recall across the 3 change scenarios are summarised in Figures A.8. For values of $\phi \leq 0.5$ DeCAFS has comparable performances to the ones of the model where we have dependence across segment. Throughout DeCAFS tends to perform similarly to or better than AR1Seg.

Figure A.9 shows a comparison between LAVA and DeCAFS when the mean is sinusoidal with abrupt jumps.

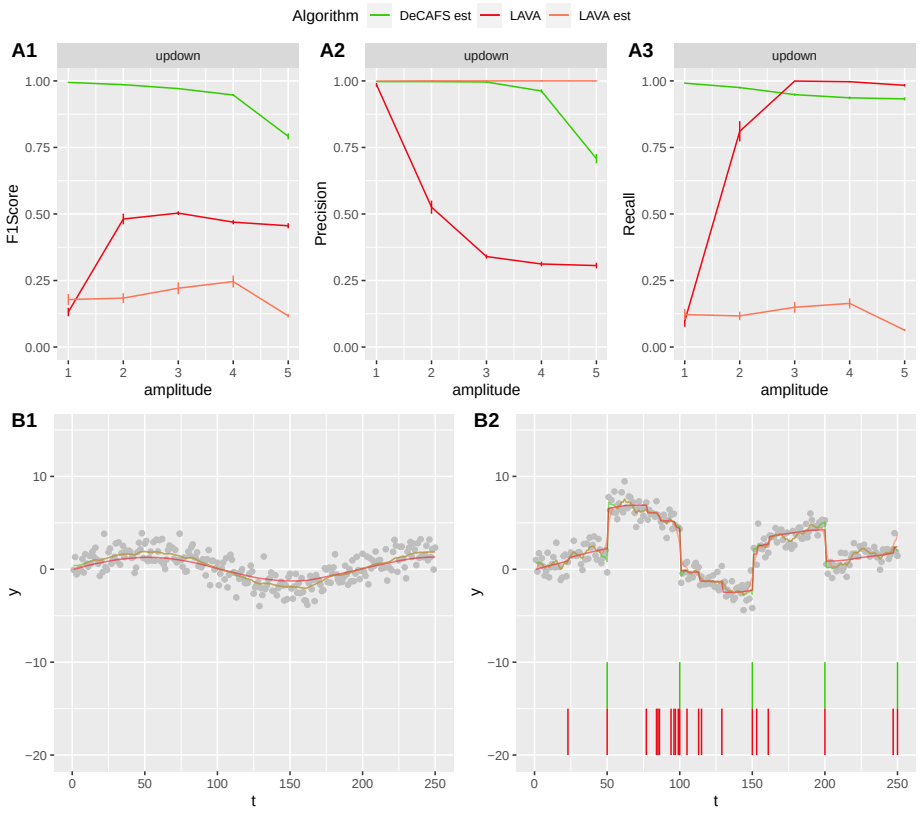


Figure A.9: On top: comparison of the F1 Score, in **A1**, Precision in **A2** and Recall, in **A3**, for DeCAFS est (in light green) and LAVA (red) and LAVA est (in orange) on the updown scenario for a sinusoidal signal over a range of different amplitudes. On the bottom the first 250 observations of two realization of the experiment with an amplitude of 2, in **B1** with no changes, whilst in **B2** with 20 changes. The continuous line over the data points represent the relative signal estimations of DeCAFS est LAVA oracle, and LAVA est; the segments their changepoint locations estimates. In **B1**, in particular, LAVA est and DeCAFS est have an almost equal signal estimation.

Appendix B

FOCuS

B.1 Proof of Proposition 7

It is straightforward to show, for example by induction, that the solution $Q_n(\mu)$ to recursion (4.7) can be written in the form

$$Q_n(\mu) = \max_{\tau=1,\ldots,n} \left\{ \sum_{i=\tau}^{n} \mu \left(x_i - \frac{\mu}{2} \right) \right\}.$$

Hence

$$\begin{aligned} \max_{\mu} Q_n(\mu) &= \max_{\tau=1,\ldots,n} \left\{ \max_{\mu} \sum_{i=\tau}^{n} \mu \left(x_i - \frac{\mu}{2} \right) \right\} \\ &= \max_{\tau=1,\ldots,n} \left\{ \frac{1}{2} \sum_{i=\tau}^{n} \left(\frac{\sum_{j=\tau}^{n} x_j}{n-\tau+1} \right)^2 \right\} \\ &= \max_{w=1,\ldots,n} \left\{ \frac{1}{2} w \left(\frac{\sum_{j=n-w+1}^{n} x_j}{w} \right)^2 \right\} \end{aligned}$$

The second line uses the fact that the maximum over μ is when μ is the sample mean of $x_{\tau:n}$. The terms in the final expression are just $(1/2)M_w(n)^2$ as required. The result in terms of $P(n)$ follows directly from $P(n) = \max_w M_w(n)$. □

B.2 Focus pseudo-code

One way to deal with the pre-change mean unknown case is to split the total cost function in two components, one for the pre-change and the post-change means. We introduce the recursion:

$$Q_n^0(\mu) = Q_{n-1}^0(\mu) + \mu(2x_n - \mu)$$
$$Q_n^1(\mu) = \max\{Q_{n-1}^1(\mu) + \mu(2x_n - \mu),\ \max_\mu Q_n^0(\mu)\},$$

where $\mathcal{Q}_n = \max_\mu Q_n^1(\mu) - \max_\mu Q_n^0(\mu)$. A description of the algorithm is found in algorithm 12. Compared to FOCuS^0 we observe a minor additional computational overhead from the need of storing, updating and maximising the Q_n^0 cost.

Algorithm 12: FOCuS (one iteration) – Pre-Change mean unknown

Data: x_t, the last realization from a data generating process X;
$Q_{n-1}^0(\mu),\ Q_{n-1}^1(\mu)$ the cost functions from the previous iteration.
Input: $\lambda > 0$
$Q_n^0(\mu) \leftarrow Q_{n-1}^0(\mu) + \mu(2x_n - \mu)$;
$Q_n^1(\mu) \leftarrow \max\{\max_\mu Q_n^0(\mu), Q_{n-1}^1(\mu) + \mu(2x_n - \mu)\}$; `// Algorithm 8`
$\mathcal{Q}_n \leftarrow \max_\mu Q_n^1(\mu) - \max_\mu Q_n^0(\mu)$; `// Theorem 5 : average` $O(\log(n))$
if $\mathcal{Q}_n \geq \lambda$ **then**
  return n *as a stopping point*;
end
return $Q_{n-1}^0(\mu),\ Q_{n-1}^1(\mu)$ *for the next iteration.*

B.3 On the expected number of changes stored by Focus

B.3.1 Variants of the FOCuS implementations

We study the number of candidate changepoints $\tau \in \{1, \cdots, n\}$ stored by FOCuS at each iteration. We first report three different variants of the FOCuS optimization introduced in Chapter 4. We can have:

- FOCuS^0 which solves the problem for a known pre-change mean μ_0 (typically 0) and unknown post-change mean μ_1.

$$\mathcal{Q}_n^0 = \max_{\substack{\tau\in\{1,\ldots,n\}\\ \mu_0=0,\mu_1\ \in\ \mathbb{R}}} \left\{ \sum_{t=1}^{\tau} (x_t - \mu_0)^2 - \sum_{t=\tau+1}^{n} (x_t - \mu_1)^2 \right\}. \tag{B.1}$$

  This problem is solved through Algorithm 8, an algorithm similar to the Melkman's algorithm (Melkman, 1987).

- FOCuS which solves the problem for both unknown pre-change and post-change means:

$$\mathcal{Q}_n = \max_{\substack{\tau\in\{1,\dots,n\}\\ \mu_1,\mu_0\in\mathbb{R}}} \left\{ \sum_{t=1}^{\tau}(x_t-\mu_0)^2 - \sum_{t=\tau+1}^{n}(x_t-\mu_1)^2 \right\}. \tag{B.2}$$

- FOCuS^{opt} which solves the problem in the case we do not know the first segment mean and the value of this mean is optimized:

$$\mathcal{Q}_n^{opt} = \max_{\substack{\tau\in\{1,\dots,n\}\\ \mu_1\in\mathbb{R}}} \left\{ \max_{\mu_0}\sum_{t=1}^{\tau}(x_t-\mu_0)^2 - \sum_{t=\tau+1}^{n}(x_t-\mu_1)^2 \right\}. \tag{B.3}$$

  This is solved in Appendix B.2. This can be done in a similar fashion to the functional-pruning update of the pDPA algorithm with one change (Rigaill, 2015).

B.3.2 Assumptions and definitions

We assume that:

$$x_i = \mu_i + \varepsilon_i, \tag{B.4}$$

where ε_i are i.i.d with a continuous distribution and μ_i is a piecewise constant signal in 1 or 2 pieces. We denote a true changepoint with τ^*.

We define the cost of a segmentation $x_{i:j}$ with a change at τ as:

$$q_{i:j,\tau}(\mu_0,\mu_1) = \sum_{t=i}^{\tau}(x_t-\mu_0)^2 + \sum_{t=\tau+1}^{j}(x_t-\mu_1)^2.$$

with pre-change and post-change means μ_0 and μ_1. As a convention, for $j=\tau$, we take: $q_{i:j,j}(\mu_0,\mu_1) = \sum_{t=i}^{j}(x_t-\mu_0)^2$.

Sets of changepoint candidates. We call $\mathcal{I}_{i:j}^0$ the set of candidate changepoints stored by FOCuS^0, $\mathcal{I}_{i:j}$ the set stored by FOCuS, and $\mathcal{I}_{i:j}^{Opt}$ the one stored by FOCuS^{opt}. By definition those will be:

$$\begin{aligned}
\mathcal{I}_{i:j}^0 &= \{\tau \mid \exists\ \mu_1, \quad \forall\tau'\neq\tau, \quad q_{i:j,\tau}(0,\mu_1) < q_{i:j,\tau'}(0,\mu_1)\}, \\
\mathcal{I}_{i:j} &= \{\tau \mid \exists\ \mu_0,\mu_1, \forall\tau'\neq\tau, \quad q_{i:j,\tau}(\mu_0,\mu_1) < q_{i:j,\tau'}(\mu_0,\mu_1)\}, \\
\mathcal{I}_{i:j}^{Opt} &= \left\{\tau \mid \exists\ \mu_1, \quad \forall\tau'\neq\tau,\ \min_{\mu_0} q_{i:j,\tau}(\mu_0,\mu_1) < \min_{\mu_0} q_{i:j,\tau'}(\mu_0,\mu_1)\right\}.
\end{aligned}$$

We will first show that $\mathcal{I}^0_{i:j}$ and $\mathcal{I}^{Opt}_{i:j}$ are in $\mathcal{I}_{i:j}$. Our goal will then be to control the size of the set $\mathcal{I}_{i:j}$ of candidate changepoints stored by FOCuS . To this end, we will consider separately the sets of positive and negative changes:

$$\mathcal{I}^+_{i:j} = \{\tau \mid \exists\ \mu_0 < \mu_1, \forall \tau' \neq \tau,\ q_{i:j,\tau}(\mu_0,\mu_1) < q_{i:j,\tau'}(\mu_0,\mu_1)\}\,. \tag{B.5}$$

$$\mathcal{I}^-_{i:j} = \{\tau \mid \exists\ \mu_0 > \mu_1, \forall \tau' \neq \tau,\ q_{i:j,\tau}(\mu_0,\mu_1) < q_{i:j,\tau'}(\mu_0,\mu_1)\}\,. \tag{B.6}$$

B.3.3 Main results

On the assumption of a realization from (B.4) we can get the following bound on the number of changepoints stored by FOCuS^0, FOCuS and FOCuS^{opt}.

Theorem 5 *For all $n \geq 1$*

$$E(\#\mathcal{I}^0_{1:n}) \leq E(\#\mathcal{I}_{1:n}) \quad \textit{and} \quad E(\#\mathcal{I}^{Opt}_{1:n}) \leq E(\#\mathcal{I}_{1:n})$$

For $D = 0$ we have:

$$E(\#\mathcal{I}^+_{1:n}) = E(\#\mathcal{I}^-_{1:n}) = \sum_1^n 1/t \leq (1 + log(n))$$

and

$$E(\#\mathcal{I}_{1:n}) = 2\sum_1^n 1/t \leq 2(1 + log(n)).$$

For $D = 1$ we have

$$E(\#\mathcal{I}_{1:n}) \leq 4(1 + log(n/2)).$$

Overview of the proof. The proof to this theorem relies on a combination of three lemmas, summarized here:

1. Lemma 1: $\mathcal{I}_{i:j}$ includes both $\mathcal{I}^{Opt}_{i:j}$ and $\mathcal{I}^0_{i:j}$;
2. Lemma 2: for $i \leq j < k$ we have $\mathcal{I}^+_{i:k} \subseteq \mathcal{I}^+_{i:j} \cup \mathcal{I}^+_{j+1:k}$;
3. Lemma 3: $\mathcal{I}_{i:j}$ is the set of extreme points of the sequence $S_{i:j}$;

and Lemma 4, that controls the number of extreme point of a random-walk (derived from Andersen, 1955; Abramson, 2012). The four lemmas are covered in details and proven in Appendix B.3.5.

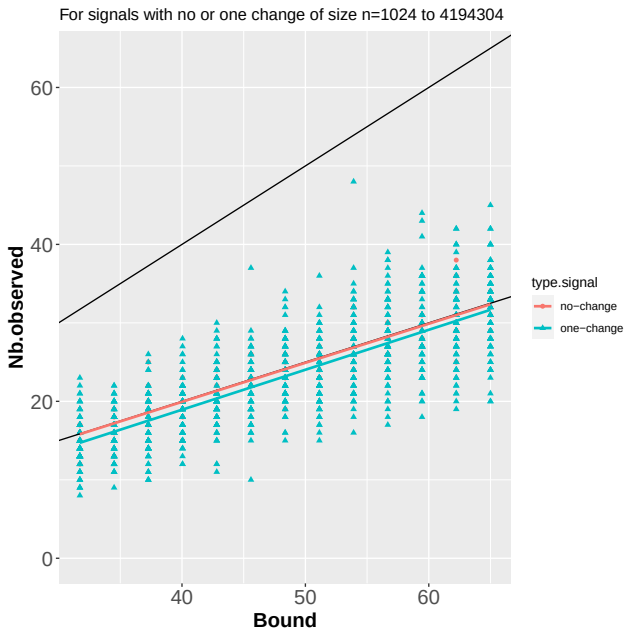


Figure B.1: Number of observed candidate stored by FOCuS for signals with no change or with one change. The two black lines represent the function $y = x$ and $y = 0.5x$. Red and blue line are the fitted regression lines for respectively the no-change and single-change scenarios.

Proof 4 *Using lemma 1 we obtain the first two inequalities.*
For $D = 0$ we apply Lemma 4 both on $\mathcal{I}^+_{1:n}$ and $\mathcal{I}^-_{1:n}$.
For $D = 1$, using lemma 2 we get that

$$\mathcal{I}^+_M(x_{1:n}) \subseteq \mathcal{I}^+_M(x_{1:\tau^*}) \cup \mathcal{I}^+_M(x_{\tau^*+1:n}).$$

We then apply lemma 4 on $\mathcal{I}^+_M(x_{1:\tau^*})$ and $\mathcal{I}^+_M(x_{\tau^*+1:n})$. The worst case is obtained for $\tau^* = n/2$. By symmetry on $\mathcal{I}^-_M(x_{1:n})$ we obtain the final result. □

B.3.4 Empirical bound evaluation

To illustrate the bound of Theorem 5 we simulate signals of various length n (from $n = 2^{10}$ to $n = 10^{22}$) without no change (5 replicates) and with one change (100 replicates). We then record the number of candidates stored by FOCuS . Where a change was present, its location was sampled uniformly between 1 and $n-1$. Similarly, the change magnitude sampled uniformly at random in $[0, 4]$. Results are summarised in figure B.1. We denote how the observed number of candidates is always less than $4(\log(n) + 1)$, with the average being around $2(\log(n) + 1)$.

B.3.5 Inclusions and convex hull Lemmas

We begin with the two inclusion lemmas.

Lemma 1

$$\mathcal{I}^0_{1:n} \subseteq \mathcal{I}_{1:n} \quad \textit{and} \quad \mathcal{I}^{Opt}_{1:n} \subseteq \mathcal{I}_{1:n}$$

Proof 5 *We get the first inclusion by definition of* $\mathcal{I}^0_{1:n}$ *and* $\mathcal{I}_{1:n}$.
Consider a change τ *in* $\mathcal{I}^{Opt}_{1:n}$. *Then there exists* μ_1 *such that for all* $\tau' \neq \tau$

$$min_{\mu_0} q_{1:n,\tau}(\mu_0, \mu_1) \leq \min_{\mu_0} q_{1:n,\tau'}(\mu_0, \mu_1).$$

Defining $\hat{\mu} = \min_{\mu_0} q_{1:n,\tau}(\mu_0, \mu_1)$, *we get:*

$$q_{i:j,\tau}(\hat{\mu}, \mu_1) = \min_{\mu_0} q_{i:j,\tau}(\mu_0, \mu_1) < \min_{\mu_0} q_{i:j,\tau'}(\mu_0, \mu_1) \leq q_{i:j,\tau'}(\hat{\mu}, \mu_1).$$

Therefore τ *is also in* $\mathcal{I}_{1:n}$. □

Lemma 2 *For* $i \leq j \leq k$

$$\mathcal{I}^+_{i:k} \quad \subseteq \quad \mathcal{I}^+_{i:j} \cup \mathcal{I}^+_{j+1:k} \tag{B.7}$$

$$\mathcal{I}^-_{i:k} \quad \subseteq \quad \mathcal{I}^-_{i:j} \cup \mathcal{I}^-_{j+1:k} \tag{B.8}$$

Proof 6 *Consider any* τ *in* $(i+1:j) \cap \mathcal{I}^+_{i:j}$, *then by definition and using equation* (B.9) *we get that*

$$\exists\ \mu_0, \mu_1, \forall\ \tau' \neq \tau, \quad q_{i:j,\tau}(\mu_0, \mu_1) < q_{i:j,\tau'}(\mu_0, \mu_1),$$

therefore τ *is also in* $\mathcal{I}^+_{i:k}$. *We proceed similarly for any* τ *in* $(j+1:k) \cap \mathcal{I}^+_{j+1:k}$. *By symmetry, we get the result for* $\mathcal{I}^-_{i:k}$. □

A useful identity For any τ, τ', μ_0 and μ_1 we have that

$$q_{i:j,\tau}(\mu_0, \mu_1) - q_{i:j,\tau'}(\mu_0, \mu_1) = (\mu_0 - \mu_1)\left(2\sum_{\tau+1}^{\tau'} x_t - \mu_0 - \mu_1\right), \tag{B.9}$$

which does not depend on i and j. This identity simplifies the proof of the following lemma which relates the set $\mathcal{I}^+_{i:j}$ and $\mathcal{I}^-_{i:j}$ to the convex hull of $S_{i:j}$.

Lemma 3 *The set of* τ *in* $\mathcal{I}^+_{i:j}$ *are the extreme points of the largest convex minorant of the sequence* $S_{i:j}$. *By symmetry, the set of* τ *in* $\mathcal{I}^-_{i:j}$ *are the extreme points of the smallest concave majorant of* $S_{i:j}$.

Proof 7 *Using equation* (B.9)*, we get that if* τ' *is in* $\mathcal{I}^+_{i:j}$ *it must be that for any* τ *and* τ''

$$\bar{x}_{\tau+1:\tau'} < \bar{x}_{\tau'+1:\tau},$$

with $\bar{x}_{\tau+1:\tau'} = \frac{\sum_{t=\tau+1}^{\tau'} x_t}{\tau'-\tau}$*. This is equivalent to for all* τ *and* τ''*:*

$$\frac{S_{\tau'} - S_\tau}{\tau' - \tau} < \frac{S_{\tau''} - S_{\tau'}}{\tau'' - \tau'},$$

and therefore is part of the largest convex minorant of the sequence $S_{i:j}$*.*

We get the result on $\mathcal{I}^-_{i:j}$ *by symmetry.* □

The next lemma is based on Andersen (1955).

Lemma 4 *Assuming the* x_t *follow an i.i.d continuous distribution on* $i:j$ *with* $\mu_t = \mu_i$ *for all* t *in* $i:j$ *then* $E(\#\mathcal{I}^+_{i:j}) = E(\#\mathcal{I}^-_{i:j}) = \sum_1^{j-i-1} 1/(t+1)$

Proof 8 *We use lemma 3 and then apply Andersen (1955) (definitions at pages 1 and 2, then pages 23-24).* □

B.4 Estimation of initial parameters

We propose a simple sequential estimator for the initial parameters needed to run FOCuS in an semi-supervised manner. The basic idea to tune the λ threshold value is to run the FOCuS procedure without a threshold on at most w values, for $w \ll n$, then compute:

$$\lambda^{(w)} = \max\{\max_{\mu}\ Q_i(\mu)\ \forall\ i = 1, 2, \dots w\}$$

Using such value as a penalty will very naively ensure that the FOCuS procedure will run for at least w observations. The next step would be the one of inflating this value $\lambda^{(w)}$ in order to provide a longer average run length to the algorithm: which is we set $\lambda = \kappa\lambda^{(w)}$. On the application within Section 4.4 simply found that the value of $\kappa = \frac{\log(w+m)}{\log(w)}(1-\hat{\sigma})$ with $w = 300,\ m = 700$ and $\hat{\sigma}$ being an estimate of the standard deviation of $y_1, \dots, y_w$ produced reasonable results.

For tuning the K parameter of the bi-weight loss, one could simply store the initial w values and, if any observation within 1.5 interquantile ranges of those values are present, *i.e.* if we are in presence of outliers, then simply tune the K to be the highest occurring value within this limit. Then, one can simply run FOCuS as described above to tune the λ parameter with the newly found K. In total, this estimation adds a linear in w computational overhead, which consist in temporarily storing the initial w values and performing the estimation of the K.

Appendix C

NUNC

C.1 Proof of Proposition 6

Proof 9 *In order to prove the desired bound, we focus on the extreme case where either* $x_{1:\tau} < q$ *and* $x_{\tau+1:W} > q$*, or vice versa, and note that this case maximises the expression*

$$\mathcal{L}(x_{t-W+1:\tau}; q) + \mathcal{L}(x_{\tau+1:t}; q) - \mathcal{L}(x_{t-W+1:t}; q), \tag{C.1}$$

for any quantile q*.*

By writing out the likelihoods in the above equation, it can be observed that $\mathcal{L}(x_{t-W+1:\tau}; q) = 0$*,* $\mathcal{L}(x_{\tau+1:t}; q) = 0$*, and*

$$\mathcal{L}(x_{t-W+1:t}; q) = \frac{\tau}{W} \log \frac{\tau}{W} - (W - \tau) \log \left(\frac{W - \tau}{W} \right) \tag{C.2}$$

in the case where each $x_{1:\tau} > q$ *and* $x_{\tau+1:W} < q$*. We also have*

$$\mathcal{L}(x_{t-W+1:t}; q) = -\frac{\tau}{W} \log \frac{\tau}{W} + (W - \tau) \log \left(\frac{W - \tau}{W} \right) \tag{C.3}$$

when $x_{1:\tau} < q$ *and* $x_{\tau+1:W} > q$*. Given both* $\frac{\tau}{W}$ *and* $\frac{W-\tau}{W}$ *are less than one, the log terms in equations* (C.2) *and* (C.3) *are both negative. As such, we can bound both equations* (C.2) *and* (C.3) *above by the following bound:*

$$\mathcal{L}(x_{t-W+1:t}; q) \leq -\frac{\tau}{W} \log \frac{\tau}{W} - (W - \tau) \log \left(\frac{W - \tau}{W} \right).$$

As this case dealt with the maximum of equation (C.1)*, this then means that for any quantile* q *and window of data* $x_1, \ldots, x_W$ *we have that*

$$\mathcal{L}(x_{t-W+1:\tau}; q) + \mathcal{L}(x_{\tau+1:t}; q) - \mathcal{L}(x_{t-W+1:t}; q) \leq -\frac{\tau}{W} \log \frac{\tau}{W} - (W - \tau) \log \left(\frac{W - \tau}{W} \right), \tag{C.4}$$

as required. Additionally, we note that this equation is decreasing in τ for fixed W.

We then consider the test statistic, given by equation (5.5)*. As a result of the bound in equation* (C.4) *if*

$$2K\left[-\frac{\tau}{W}\log\frac{\tau}{W}-(W-\tau)\log\left(\frac{W-\tau}{W}\right)\right]\leq K\beta$$

then detection is impossible, as the bound for the test statistic does not exceed the threshold for the test. As a result, we conclude that if

$$-\frac{\tau^*}{W}\log\frac{\tau^*}{W}-(W-\tau^*)\log\left(\frac{W-\tau}{W}\right)\leq\frac{\beta}{2}$$

then due to the fact the expression decreases as τ increases then for $\tau>\tau^*$ detection is impossible. This completes the proof.

C.2 Proof of Proposition 5.2.2

Proof 10 *A false alarm by the time t under NUNC Local can be written as*

$$\begin{aligned}&=\mathrm{P}\left(\bigcup_{s=W}^{t}\max_{s-W+1\leq\tau\leq s}\sum_{k=1}^{K}2\left[\mathcal{L}(x_{s-W+1:\tau};q_k)+\mathcal{L}(x_{\tau+1:s};q_k)-\mathcal{L}(x_{s-W+1:s};q_k)\right]\geq K\beta\right)\\&\leq\sum_{s=W}^{t}\sum_{\tau=s-W+1}^{s}\mathrm{P}\left(\sum_{k=1}^{K}2\left[\mathcal{L}(x_{s-W+1:\tau};q_k)+\mathcal{L}(x_{\tau+1:s};q_k)-\mathcal{L}(x_{s-W+1:s};q_k)\right]\geq K\beta\right)\end{aligned}\tag{C.5}$$

The next part of the proof uses the fact that, under the i.i.d. assumption, asymptotically for any quantile the following holds

$$2\left[\mathcal{L}(x_{s-W+1:\tau};q_k)+\mathcal{L}(x_{\tau+1:s};q_k)-\mathcal{L}(x_{s-W+1:s};q_k)\right]\sim\chi^2_1.$$

As in Wainwright (2019), it can be shown that if a random variable X_i follows a χ^2_1 distribution then it is Sub-Exponential with parameters 4 and 4. That is, $X_i\sim\mathrm{SE}(4,4)$.

Under dependence, as is the case between different quantiles, if $X_i\sim\mathrm{SE}(\nu_i^2,b_i)$ then $\sum_{i=1}^{n}X_i-\mathrm{E}(X_i)\sim\mathrm{SE}\left(\left(\sum_{i=1}^{n}\nu_i\right)^2,\max_i b_i\right)$, and so

$$\sum_{k=1}^{K}\left[\mathcal{L}(x_{s-W+1:\tau};q_k)+\mathcal{L}(x_{\tau+1:s};q_k)-\mathcal{L}(x_{s-W+1:s};q_k)\right]-K\sim\mathrm{SE}(4K^2,4).$$

We then use a well known bound on the subexponential tail (Vershynin, 2018) to obtain

$$\begin{aligned}
&\sum_{s=W}^{t}\sum_{\tau=s-W+1}^{s} \mathrm{P}\left(\sum_{k=1}^{K} 2\left[\mathcal{L}(x_{s-W+1:\tau};q_k)+\mathcal{L}(x_{\tau+1:s};q_k)-\mathcal{L}(x_{s-W+1:s};q_k)\right]\geq K\beta\right)\\
&=\sum_{s=W}^{t}\sum_{\tau=s-W+1}^{s} \mathrm{P}\left(\sum_{k=1}^{K} 2\left[\mathcal{L}(x_{s-W+1:\tau};q_k)+\mathcal{L}(x_{\tau+1:s};q_k)-\mathcal{L}(x_{s-W+1:s};q_k)\right]-K\geq K\beta-K\right)\\
&\leq W(t-W+1)\exp\left(-\frac{1}{2}\min\left\{\frac{K\beta-K}{4},\frac{(K\beta-K)^2}{4K^2}\right\}\right). \qquad \text{(C.6)}
\end{aligned}$$

We can set this final line equal to α to control our desired false alarm rate. We have two cases in equation (C.6) *and must bound above by the largest of these. The first case is that*

$$W(t-W+1)\exp\left(-\frac{K\beta_1-K}{8}\right)=\alpha$$

in which case we choose

$$\beta_1=1-8K^{-1}\log\left(\frac{\alpha}{W(t-W+1)}\right).$$

On the other hand we have the case where

$$W(t-W+1)\exp\left(-\frac{(K\beta_2-K)^2}{8K^2}\right)=\alpha$$

and solving this gives

$$\beta_2=1+2\sqrt{2\log\left(\frac{W(t-W+1)}{\alpha}\right)}.$$

We then choose the larger of β_1 and β_2, completing the proof.

Bibliography

Abramson, J. S. (2012). *Some minorants and majorants of random walks and Lévy processes*. PhD thesis, UC Berkeley.

Ahmad, S., Lavin, A., Purdy, S., and Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147.

Alvarez-Montoya, J., Carvajal-Castrillón, A., and Sierra-Pérez, J. (2020). In-flight and wireless damage detection in a uav composite wing using fiber optic sensors and strain field pattern recognition. *Mechanical Systems and Signal Processing*, 136:106526.

Andersen, E. S. (1955). On the fluctuations of sums of random variables ii. *Mathematica Scandinavica*, pages 195–223.

Baranowski, R., Chen, Y., and Fryzlewicz, P. (2016). Narrowest-over-threshold detection of multiple change-points and change-point-like features. *arXiv preprint arXiv:1609.00293*.

Bardwell, L., Fearnhead, P., Eckley, I. A., Smith, S., and Spott, M. (2019). Most recent changepoint detection in panel data. *Technometrics*, 61(1):88–98.

Barry, D. and Hartigan, J. A. (1993). A bayesian analysis for change point problems. *Journal of the American Statistical Association*, 88(421):309–319.

Basseville, M., Benveniste, A., Goursat, M., and Meve, L. (2007). In-flight vibration monitoring of aeronautical structures. *IEEE Control Systems Magazine*, 27(5):27–42.

Bauschke, H. H. and Combettes, P. L. (2011). *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer.

Bleakley, K. and Vert, J.-P. (2011). The group fused lasso for multiple change-point detection. *arXiv preprint arXiv:1106.4199*.

Chakar, S., Lebarbier, E., Lévy-Leduc, C., and Robin, S. (2017). A robust approach for estimating change-points in the mean of an AR(1) process. *Bernoulli*, 23(2):1408–1447.

Chakraborti, S. and van de Wiel, M. A. (2008). A nonparametric control chart based on the mann-whitney statistic. *Institute of Mathematical Statistics Collections*, page 156–172.

Chang, S.-T. and Lu, K.-P. (2016). Change-point detection for shifts in control charts using em change-point algorithms. *Quality and Reliability Engineering International*, 32(3):889–900.

Chen, H. (2019). Sequential change-point detection based on nearest neighbors. *Ann. Statist.*, 47(3):1381–1407.

Chen, Z. and Tian, Z. (2010). Modified procedures for change point monitoring in linear models. *Mathematics and computers in simulation*, 81(1):62–75.

Chernozhukov, V., Hansen, C., and Liao, Y. (2017). A lava attack on the recovery of sums of dense and sparse signals. *The Annals of Statistics*.

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*, pages 475–507.

Chu, C.-S. J., Hornik, K., and Kaun, C.-M. (1995). Mosum tests for parameter constancy. *Biometrika*, 82(3):603–617.

Clifford, G. D., Silva, I., Moody, B., Li, Q., Kella, D., Shahin, A., Kooistra, T., Perry, D., and Mark, R. G. (2015). The physionet/computing in cardiology challenge 2015: reducing false arrhythmia alarms in the icu. In *2015 Computing in Cardiology Conference (CinC)*, pages 273–276. IEEE.

Coelho, M., Graham, M., and Chakraborti, S. (2017). Nonparametric signed-rank control charts with variable sampling intervals. *Quality and Reliability Engineering International*, 33(8):2181–2192.

Cui, Y., Ahmad, S., and Hawkins, J. (2016). Continuous online sequence learning with an unsupervised neural network model. *Neural computation*, 28(11):2474–2504.

Dette, H. and Gösmann, J. (2020). A likelihood ratio approach to sequential change point detection for a general class of parameters. *Journal of the American Statistical Association*, 115(531):1361–1377.

Eco, U. (1995). *How to travel with a salmon: and other essays*, chapter How to write an introduction. HMH.

Eiauer, P. and Hackl, P. (1978). The use of mosums for quality control. *Technometrics*, 20(4):431–436.

Eichinger, B. and Kirch, C. (2018). A mosum procedure for the estimation of multiple random change points. *Bernoulli*, 24(1):526–564.

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

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

Fearnhead, P., Maidstone, R., and Letchford, A. (2018). Detecting changes in slope with an l0 penalty. *Journal of Computational and Graphical Statistics*, pages 1–11.

Fearnhead, P. and Rigaill, G. (2019). Changepoint detection in the presence of outliers. *Journal of the American Statistical Association*, 114(525):169–183.

Fisch, A., Bardwell, L., and Eckley, I. A. (2020). Real time anomaly detection and categorisation. *arXiv preprint arXiv:2009.06670*.

Fisch, A., Eckley, I. A., and Fearnhead, P. (2018). A linear time method for the detection of point and collective anomalies. *arXiv preprint arXiv:1806.01947*.

Fisch, A., Eckley, I. A., and Fearnhead, P. (2019). Subset multivariate collective and point anomaly detection. *arXiv preprint arXiv:1909.01691*.

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

Fridman, P. (2010). A method of detecting radio transients. *Monthly Notices of the Royal Astronomical Society*, 409(2):808–820.

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

Fryzlewicz, P. (2018). Tail-greedy bottom-up data decompositions and fast multiple change-point detection. *The Annals of Statistics*, 46(6B):3390–3421.

Fryzlewicz, P. (2020). Detecting possibly frequent change-points: wild binary segmentation 2 and steepest-drop model selection. *Journal of the Korean Statistical Society*, 49(4):1027–1070.

Fryzlewicz, P. and Rao, S. S. (2014). Multiple-change-point detection for auto-regressive conditional heteroscedastic processes. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, pages 903–924.

Fuschino, F., Campana, R., Labanti, C., Evangelista, Y., Feroci, M., Burderi, L., Fiore, F., Ambrosino, F., Baldazzi, G., Bellutti, P., et al. (2019). Hermes: An ultra-wide band x and gamma-ray transient monitor on board a nano-satellite constellation. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 936:199–203.

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

Gordon, L. and Pollak, M. (1994). An efficient sequential nonparametric scheme for detecting a change of distribution. *The Annals of Statistics*, 22(2):763 – 804.

Gösmann, J., Kley, T., and Dette, H. (2019). A new approach for open-end sequential change point monitoring. *arXiv preprint arXiv:1906.03225*.

Green, P. J. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732.

Harchaoui, Z. and Lévy-Leduc, C. (2007). Catching change-points with lasso. In *NIPS*, volume 617, page 624.

Hawkins, D. M. and Deng, Q. (2010). A nonparametric change-point control chart. *Journal of Quality Technology*, 42(2):165–173.

Haynes, K., Eckley, I. A., and Fearnhead, P. (2017a). 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. (2017b). A computationally efficient nonparametric approach for changepoint detection. *Statistics and Computing*, 27(5):1293–1305.

Hinkley, D. V. (1971). Inference about the change-point from cumulative sum tests. *Biometrika*, 58(3):509–523.

Hocking, T. D., Rigaill, G., Fearnhead, P., and Bourque, G. (2017). A log-linear time algorithm for constrained changepoint detection. *arXiv preprint arXiv:1703.03352*.

Hocking, T. D., Rigaill, G., Fearnhead, P., and Bourque, G. (2020). Constrained dynamic programming and supervised penalty learning algorithms for peak detection in genomic data. *Journal of Machine Learning Research*.

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

Huber, P. J. (2004). *Robust statistics*, volume 523. John Wiley & Sons.

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

Jalali, A., Ravikumar, P., and Sanghavi, S. (2013). A dirty model for multiple sparse regression. *IEEE Transactions on Information Theory*, 59(12):7947–7968.

Jeske, D. R., Stevens, N. T., Tartakovsky, A. G., and Wilson, J. D. (2018). Statistical methods for network surveillance. *Applied Stochastic Models in Business and Industry*, 34(4):425–445.

Jewell, S. W., Hocking, T. D., Fearnhead, P., and Witten, D. M. (2020). Fast nonconvex deconvolution of calcium imaging data. *Biostatistics*, 21(4):709–726.

Killick, R., Eckley, I. A., Ewans, K., and Jonathan, P. (2010). Detection of changes in variance of oceanographic time-series using changepoint analysis. *Ocean Engineering*, 37(13):1120–1126.

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598.

Kim, C.-J., Morley, J. C., and Nelson, C. R. (2005). The structural break in the equity premium. *Journal of Business & Economic Statistics*, 23(2):181–191.

Kim, S.-J., Koh, K., Boyd, S., and Gorinevsky, D. (2009). l1 trend filtering. *SIAM review*, 51(2):339–360.

Kirch, C. and Kamgaing, J. T. (2015). On the use of estimating functions in monitoring time series for change points. *Journal of Statistical Planning and Inference*, 161:25–49.

Kirch, C., Weber, S., et al. (2018). Modified sequential change point procedures based on estimating functions. *Electronic Journal of Statistics*, 12(1):1579–1613.

Korkas, K. K. and Fryzlewicz, P. (2017). Multiple change-point detection for non-stationary time series using wild binary segmentation. *Statistica Sinica*, pages 287–311.

Kovács, S., Li, H., Bühlmann, P., and Munk, A. (2020). Seeded binary segmentation: A general methodology for fast and optimal change point detection. *arXiv preprint arXiv:2002.06633*.

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.

Leonardi, F. and Bühlmann, P. (2016). Computationally efficient change point detection for high-dimensional regression. *arXiv preprint arXiv:1601.03704*.

Lin, K., Sharpnack, J. L., Rinaldo, A., and Tibshirani, R. J. (2017). A sharp error analysis for the fused lasso, with application to approximate changepoint screening. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6884–6893. Curran Associates, Inc.

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

Liu, L., Zi, X., Zhang, J., and Wang, Z. (2013). A sequential rank-based nonparametric adaptive ewma control chart. *Communications in Statistics - Simulation and Computation*, 42(4):841–859.

Maidstone, R., Hocking, T., Rigaill, G., and Fearnhead, P. (2017). On optimal multiple changepoint algorithms for large data. *Statistics and Computing*, 27(2):519–533.

Meier, A., Kirch, C., and Cho, H. (2021). mosum: A package for moving sums in change-point analysis. *Journal of Statistical Software*, 97(8):1–42.

Melkman, A. A. (1987). On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25(1):11–12.

Meng, X.-L. and Rubin, D. B. (1993). Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278.

Mood, A. M. (1954). On the asymptotic efficiency of certain nonparametric two-sample tests. *The Annals of Mathematical Statistics*, pages 514–522.

Mukherjee, A. and Chakraborti, S. (2012). A distribution-free control chart for the joint monitoring of location and scale. *Quality and Reliability Engineering International*, 28(3):335–352.

Murakami, H. and Matsuki, T. (2010). A nonparametric control chart based on the mood statistic for dispersion. *The International Journal of Advanced Manufacturing Technology*, 49:757–763.

Nicolas, P., Leduc, A., Robin, S., Rasmussen, S., Jarmer, H., and Bessières, P. (2009). Transcriptional landscape estimation from tiling array data using a model of signal shift and drift. *Bioinformatics*, 25(18):2341–2347.

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*, 5:557–572.

Padilla, O. H. M., Athey, A., Reinhart, A., and Scott, J. G. (2019). Sequential nonparametric tests for a change in distribution: An application to detecting radiological anomalies. *Journal of the American Statistical Association*, 114(526):514–528.

Page, E. (1955). A test for a change in a parameter occurring at an unknown point. *Biometrika*, 42(3/4):523–527.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, 41(1/2):100–115.

Peng, T., Leckie, C., and Ramamohanarao, K. (2004). Proactively detecting distributed denial of service attacks using source ip address monitoring. In *International conference on research in networking*, pages 771–782. Springer.

Pepelyshev, A. and Polunchenko, A. (2017). Real-time financial surveillance via quickest change-point detection methods. *Statistics and its interface*, 10:93–106.

Picard, F., Lebarbier, É., Budinská, E., and Robin, S. (2011). Joint segmentation of multivariate gaussian processes using mixed linear models. *Computational Statistics & Data Analysis*, 55(2):1160–1170.

Popescu, T. D. and Aiordăchioaie, D. (2017). New procedure for change detection operating on rényi entropy with application in seismic signals processing. *Circuits, Systems, and Signal Processing*, 36(9):3778–3798.

Pouliezos, A. and Stavrakakis, G. S. (2013). *Real time fault monitoring of industrial processes*, volume 12. Springer Science & Business Media.

Raftery, A. E. and Akman, V. (1986). Bayesian analysis of a poisson process with a change-point. *Biometrika*, pages 85–89.

Reckrühm, K. (2019). Estimating multiple structural breaks in time series-a generalized mosum approach based on estimating functions.

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.

Reynolds, M. R. (1975). Approximations to the average run length in cumulative sum control charts. *Technometrics*, 17(1):65–71.

Rigaill, G. (2010). Pruned dynamic programming for optimal multiple change-point detection. *arXiv preprint arXiv:1004.0887*, 17.

Rigaill, G. (2015). A pruned dynamic programming algorithm to recover the best segmentations with 1 to kmax change-points. *Journal de la Societe Francaise de Statistique*, 156(4):180–205.

Romano, G., Rigaill, G., Runge, V., and Fearnhead, P. (2021). Detecting abrupt changes in the presence of local fluctuations and autocorrelated noise. *Journal of the American Statistical Association*, 0(0):1–16.

Ross, G. J. (2021). Nonparametric detection of multiple location-scale change points via wild binary segmentation. *arXiv preprint arXiv:2107.01742*.

Ross, G. J. 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.

Ruanaidh, J. J. O. and Fitzgerald, W. J. (2012). *Numerical Bayesian methods applied to signal processing*. Springer Science & Business Media.

Runge, V. (2020). Is a finite intersection of balls covered by a finite union of balls in euclidean spaces? *Journal of Optimization Theory and Applications*, 187(2):431–447.

Runge, V., Hocking, T. D., Romano, G., Afghah, F., Fearnhead, P., and Rigaill, G. (2020a). gfpop: an r package for univariate graph-constrained change-point detection. *arXiv preprint arXiv:2002.03646*.

Runge, V., Pascucci, M., and de Boishebert, N. D. (2020b). Change-in-slope optimal partitioning algorithm in a finite-size parameter space. *arXiv preprint arXiv:2012.11573*.

Safikhani, A. and Shojaie, A. (2020). Joint structural break detection and parameter estimation in high-dimensional non-stationary var models. *Journal of the American Statistical Association*, (just-accepted):1–26.

Scott, A. J. and Knott, M. (1974). A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 30(3):507–512.

Sen, A. and Srivastava, M. S. (1975). On tests for detecting change in mean. *The Annals of statistics*, pages 98–108.

Siegmund, D. (2013). Change-points: From sequential detection to biology and back. *Sequential Analysis*, 32(1):2–14.

Siems, T., Hellmuth, M., and Liebscher, V. (2019). Simultaneous credible regions for multiple changepoint locations. *Journal of Computational and Graphical Statistics*, 28(2):290–298.

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., Nikiforov, I., and Basseville, M. (2014). *Sequential Analysis: Hypothesis Testing and Changepoint Detection*. CRC Press.

Tartakovsky, A. G., Polunchenko, A. S., and Sokolov, G. (2012). Efficient computer network anomaly detection by changepoint detection methods. *IEEE Journal of Selected Topics in Signal Processing*, 7(1):4–11.

Tartakovsky, A. G. and Rozovskiĭ, B. L. (2007). A nonparametric multichart cusum test for rapid intrusion detection. In *in: Proc. Joint Statistical Meetings, 7–11 August 2007*, volume 7, page 11. Citeseer.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108.

Tibshirani, R. J. et al. (2014). Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics*, 42(1):285–323.

Tickle, S., Eckley, I., Fearnhead, P., and Haynes, K. (2020). Parallelization of a common changepoint detection method. *Journal of Computational and Graphical Statistics*, 29(1):149–161.

Vershynin, R. (2018). *High-dimensional probability: an introduction with applications in data science*, page 37. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Wainwright, M. (2019). *High-dimensional statistics: a non-asymptotic viewpoint*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Wang, D., Zhang, L., and Xiong, Q. (2017). A non parametric cusum control chart based on the mann–whitney statistic. *Communications in Statistics - Theory and Methods*, 46(10):4713–4725.

Wang, T. and Samworth, R. J. (2018). High dimensional change point estimation via sparse projection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(1):57–83.

Wilks, S. S. (1938). The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9(1):60 – 62.

Xie, L., Xie, Y., and Moustakides, G. V. (2019). Asynchronous multi-sensor change-point detection for seismic tremors. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 787–791. IEEE.

Yao, Y.-C. (1988). Estimating the number of change-points via Schwarz's criterion. *Statistics & Probability Letters*, 6(3):181–189.

Yao, Y.-C. and Au, S.-T. (1989). Least-squares estimation of a step function. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 370–381.

Yu, Y., Padilla, O. H. M., Wang, D., and Rinaldo, A. (2020). A note on online change point detection. *arXiv preprint arXiv:2006.03283*.

Zou, C., Yin, G., Feng, L., Wang, Z., et al. (2014). Nonparametric maximum likelihood approach to multiple change-point problems. *The Annals of Statistics*, 42(3):970–1002.