Offline change point detection for binary data via regularization methods

Lucas de Oliveira Prates

Dissertação/Tese apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Estatística

Programa: Estatística

Orientador: Prof[a]. Florencia Graciela Leonardi

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES

São Paulo, fevereiro de 2021

Offline change point detection for binary data

Esta é a versão original da dissertação/tese elaborada pelo
candidato Lucas de Oliveira Prates, tal como
submetida à Comissão Julgadora.

Agradecimentos

Apesar de levar meu nome, este trabalho não seria possível sem o apoio de diversas pessoas. Desta forma, gostaria de agradecer brevemente à algumas delas. Agradeço à meus pais Liana e Geraldo, meu irmão Thiago, e minha amiga Silvana, pelo amor, carinho e confiança ao longo de minha vida. Acima de tudo, agraço por terem me privilegiado com a oportunidade estudar.

Agradeço à minha orientadora, Florencia, pela sua paciência, confiança, e direcionamento na pesquisa. Agradeço à meus amigos de infância e de faculdade, com quem compartilhei memórias queridas.

Por fim, sou grato à todos que contribuiram com o conhecimento humano até então acumulado, especialmente em matemática, estatística e computação. O meu aprendizado é fruto do esforço coletivo de todas essas pessoas.

Resumo

Prates, L. O. **Detecção offline de pontos de mudança para dados binários via métodos de regularização**. 2021. 120 f. Dissertação - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

Em análise de séries temporais, o problema de detecção de pontos de mudança consiste em estimar os tempos nos quais a distribuição de probabilidade sofre alguma alteração. Sob à hipótese de que os dados têm distribuição Bernoulli, o problema pode ser visto como estimar os tempos nos quais o parâmetro de probabilidade se altera. Neste trabalho, apresentaremos métodos estatísticos para estimar o número e a localização dos pontos de mudança quando os dados têm distribuição Bernoulli. Os métodos escolhidos foram verossimilhança penalizada, Fused LASSO e métodos baseados em validação cruzada. Provamos a consistência de alguns dos métodos propostos, e fornecemos um estudo de simulação para comparação de modelos. Por fim, aplicamos os modelos no problema de identificação de regiões de homozigose em arrays de SNPs.

Palavras-chave: detecção de pontos de mudança, regularização, SNPs.

Abstract

Prates, L. O. **Offline change point detection for binary data via regularization methods**. 2021. 120 f. Master's Thesis - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

In time series analysis, change point detection consists in estimating the times in which the probability distribution changes. Under the assumption that the data is Bernoulli distributed, the problem can be seen as estimating the time in which the probability parameter changes. We will present statistical methods based on penalized likelihood to estimate the number and location of the change points. The chosen methods were penalized likelihood, Fused LASSO and methods based on cross validation. The consistency of some of the methods is proved, and a simulation study is provided to compare models. We then apply the models to the identification of regions of homozygosity in SNP arrays.

Keywords: change point detection, regularization, SNPs.

Contents

List of abbreviations

AIC	Akaike Information Criteria
BIC	Bayesian Information Criteria
CDF	Cumulative Distribution Function
CV	Cross Validation
CVDYNSEG	Cross Validation on regularization constant Algorithm
CVSEG	Cross Validation on the number of segments Algorithm
DPS	Dynamical Segmentation Algorithm
EPE	Expected Prediction Error
HS	Hierarhical Segmentation Algorithm
MLE	Maximum Likelihood Estimator
PL	Penalized Likelihood
PML	Penalized Maximum Likelihood

List of Figures

List of Tables

Chapter 1

Introduction

Humans observe with keen interest the processes of this dynamical universe aiming to describe and mimic them. While watching the process in its entirety, it might be only a few singularities, abrupt changes in the process states, that create a sparkle in the viewer's mind, providing fuel for his creativity. By carefully describing what these changes are and how they occur, he is able to unravel the whole. Furthermore, he could detect new changes, rapidly reacting after one occurs, or even predict when it will occur, taking actions to prevent or prepare himself.

Change point detection is a multidisciplinary field in statistics that studies changes in processes of very different natures. Framed in the context of time series analysis, it uses mathematics, statistics, and computer science to provide rigorous, reliable methodologies to detect and predict subtle and complex changes.

Grasping the concept of change is complex and context-dependent. However, this broad interpretation of what is to change allows the field to be useful to many different areas. Like any application in statistics, defining the concepts correctly and preparing the data might be more important than the method used.

In this Chapter, we begin by posing the basic idea behind change point detection. Then, we discuss some aspects of the problem considered in research and applications.

1.1 Problem Definition

Let $\mathcal{T}$ be an ordered set and consider we have a sequence of $\{X_{t_i}\}_{i=1}^{m}$, $m \in \mathbb{N}$ independent random variables where X_{t_i} has CDF F_i, $t_i \in \mathcal{T}$. Defining $1 : (m-1) = \{1, \ldots, m-1\}$, we are interested in obtaining the change point set C^* given by

$$C^* = \{c \in 1 : (m-1) | F_c \neq F_{c+1}\} \quad .$$

We can see this as blocks of random variables with the same distribution. Our task is to detect the blocks and the CDF of each block. The number of change points can either be known or unknown, resulting in very different approaches. We can also have multiple sequences of data, as will be the case in this work.

For the parametric approach, we usually define the CDFs as being elements of a family $\mathcal{F} = \{F_\theta : \theta \in \Theta\}$, where the CDF is fully specified by the parameter θ. In this case, the problem translates to detecting when the changes in the parameter. For example, if the data comes from a Gaussian distribution with unknown mean and variance, then we could design methods to estimate the locations in which the mean changes.

1.2 Aspects of change point problems

Changes have multiple aspects to consider, and it might not be possible to devise a method that works irrespective of the problem. Here, we will discuss the possible scenarios briefly. The next

section introduces the most common mathematical formulation of the problem.

The first and most clear distinction is on single and multiple change point problems. A single change point problem is when we have at most one change, and our task is only to find its location if the change exists. This was the first studied model, presented on the pioneering work of Page [1954].

A multiple change point problem requires us to detect the location of possible multiple change points. Two different settings are possible here: known and unknown number of change points. For the second problem, the number of points may range from no change points at all up to a maximum number of change points established beforehand. This difficulty usually deteriorates the performance and run time, and it is much harder to prove their correctness.

Albeit most of the applications consider change points with respect to time, many methods are general enough to work with other variables and dimensions where an order relation is available. For instance, when analyzing a SNP array, we search change points in the base pairs' physical position, not on a time dimension.

Another important distinction is between online and offline change point detection. In online problems, we expect to continually receive new data to analyze in the dimension of interest. This type of problem is common in economics, quality control, and signal processing, and we can have many different objectives. We could try to react as fast as possible when noticing a change, predict the next change, estimate how long the current state will last, and so on.

In offline problems, also called retrospective analysis, we are interested in detecting changes in a phenomenon that already occurred or that has a fixed size. Again, there is a broad range of interesting questions that can help us better capture what happened. Can we associate the change with some other event known to have occurred near the change point estimated?

If we dive into mathematical and statistical assumptions, we can split the research further. Lee [2010] provides a comprehensive list on change point detection research up to 2010, summarising the number of papers considering the various problems and methodologies discussed.

Parametric and nonparametric settings have been advanced to solve problems such as detecting changes in the mean, variance and regression slope. Parametric change point detection was introduced by Page [1955], studying the Normal mean change point model. Chen and Gupta [2011] offers an introduction to parametric change point detection with a mathematically rigorous approach, presenting well-studied change point models such as changes in Normal mean, Normal variance, Poisson rate, binomial parameter, hazard rate, while providing applications in different areas.

A forking point between researches is the estimation approach. Bayesian, maximum likelihood, and regularizations estimation methods have been applied to online and offline problems, sometimes considering different aspects of the problem.

Regularized estimators are often used when the number of change points is unknown. Zou et al. [2014] used the BIC criterion for estimating change points combined with a nonparametric maximum likelihood approach. More recently, sparsity inducing regularizations have received more attention, with Levy-leduc and Harchaoui [2008] introducing the use of Lasso for change point detection.

For bayesian methods, the works of Raftery and Akman [1986] and Carlin et al. [1992] study offline single change point detection, modeling different aspects of the problem. In online problems, Adams and MacKay [2007] modeled the sequence run length, i.e., the time since the last change. An extension of the model by Agudelo-España et al. [2020] also models residual time, i.e., the time until the next change point.

Other techniques, such as kernel-based methods and sliding windows, have also been considered. Harchaoui and Cappe [2007] presents the use of kernels for offline change point detection, and Bouchikhi et al. [2020] investigates the usage of a kernel-based algorithm for online problems.

In practice, most estimators proposed are not so easily computed. Frequently, the estimators are formulated as solutions to optimization problems. Different paradigms and algorithms are considered for calculations. Usually, a there is a trade-off: use exact but more time-consuming search methods or fast greed search methods for an approximation. The choice usually depends on the aspects of the problem at hand. Chapter 3 discusses some paradigms and algorithms that can be applied to

change point detection.

1.3 Applications

Since change point detection is so multidisciplinary and broad, selective reviews aimed at the statistical community have been published. Niu et al. [2016] shows classical and modern applications of change point detection, and poses the problem mathematically for different settings. Truong et al. [2020] presents a more thorough discussion, giving a complete description of a broad approach to change point problems. Here, we discuss briefly some real problems in which change point detection methods were applied.

Based on the pioneering work of Page [1954], the well known CUSUM is one of the widest studied method applied to several fields. Williams et al. [1992] presents a first application of the CUSUM to medicine, and the work of Li et al. [2018] expands the method for data stream anomaly detection and apply it to industrial data.

An early application of change point detection in meteorology is given in Cobb [1978], which works on the single change point problem using maximum likelihood and apply his results to the Nile River data set. The data consists of annual volume measures of the Nile River at Aswan, from 1871 to 1970, and the goal was to understand if there was an abrupt change in rainfall regime near the beginning of the last century. Reeves et al. [2007] provides a review on change point methods applied to climate change.

With the exponential growth of computer power, applications using genomics, videos, and audios data sources are becoming available and feasible. In genetics, Castro et al. [2018] uses a regularized approach to identify recombination hotspots using SNP arrays, regions of the chromosome with higher recombination rates. Celisse et al. [2018] uses kernel-based methods to identify DNA copy number alterations, which have been associated with diseases in humans.

Tahmasbi and Rezaei [2008] presents a change point model for GARCH models in speech recognition tasks, trying to identify intervals of speech and non-speech. Application on satellite image time series was provided by Verbesselt et al. [2010], investigating the land cover variation over time.

As examples of online problems, Agudelo-España et al. [2020] uses their online bayesian model to monitor reliably sleep stages using EEG/EMG data. Tartakovsky et al. [2006] shows that change point detection can be used in cybersecurity to identify attacks on networks using network traffic data.

The examples provided are only a few in a range of possibilities of the applications of change point detection. However, there is a critique that the performance of most methods is evaluated on simulated data or on small time series data sets with unreliable ground truth. To assess that problem, van den Burg and Williams [2020] created a data designed to evaluate change point detection algorithms. They also run several models and present a benchmark, comparing the performance of the methods.

Chapter 2

Statistical Model and Estimation

This chapter describes the statistical model for the offline change point detection for binary data, proposes estimation methods and states the main theorems of this work.

2.1 Statistical Model

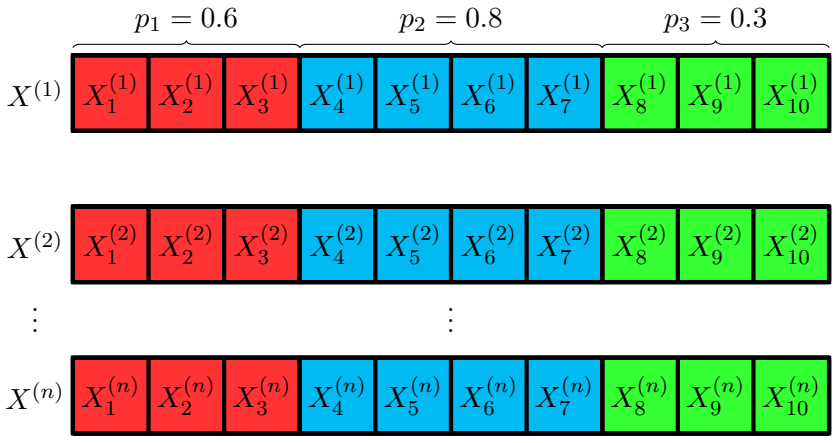
Let m be a positive integer and consider a random vector $\mathbf{X} = (X_1, \ldots, X_m)$ taking values in $\{0,1\}^m$. Given two integers r and s, with $r \leq s$, we use the notation $r : s$ for the set $\{r, r+1 \ldots, s\}$. Let $C = \{c_1, \ldots, c_k\} \subseteq 1 : (m-1)$ be an ordered subset of size k, and define $c_0 = 0$ and $c_{k+1} = m$. We say that C is a change point set for $\mathbf{X}$ if, for all j in $0 : k$, we have that $(X_{1+c_j}, \ldots, X_{c_{j+1}})$ are i.i.d. with distribution **Bernoulli**(p_j), with $p_j \neq p_{j+1}$ for all j, and are independent of the random variables from other blocks.

More formally, write the parameter space as $\Theta = \{(C, \mathbf{p}) : C \subseteq 1 : (m-1), \mathbf{p} \in [0,1]^{|C|+1}\}$. For $\theta = (C, \mathbf{p}) \in \Theta$, the probability function of $\mathbf{X}$ is

$$\mathbb{P}_\theta(\mathbf{X}) = \prod_{j=0}^{k} \prod_{c=1+c_j}^{c_{j+1}} (1-p_j)^{(1-X_c)} p_j^{X_c} \quad .$$

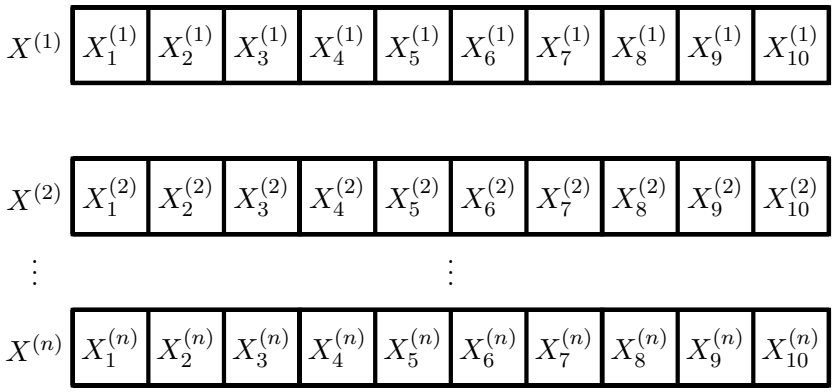
The probability function has a simple closed expression, and hence approaches using the likelihood function are straightforward.

Example 2.1.1. We provide a simple example with an illustration to aid in the comprehension of the model. Suppose we have n i.i.d. samples drawn from our model where $m = 10$, $C = \{3, 7\}$ and $\mathbf{p} = (0.6, 0.8, 0.3)$. The figure below represents samples from this setting.



Each color represents a block, and each variable of that color has Bernoulli distribution with the block parameter. All random variables are independent of each other. However, this colorful

picture can only be drawn if we know the parameters that generated the data. In practice, we get a sample like the one in the figure below.



We do not know how many blocks there are, the change points locations, or the probabilities, and we would like to estimate these quantities.

2.2 Estimation

Given n i.i.d. samples $\mathbf{X} = \{\mathbf{X}^{(i)}\}_{i=1}^n$ our interest is in estimating the change point set C^*, the number of change points $|C^*| = k^*$, and the probability vector of the blocks $\mathbf{p}^*$, where (C^*, p^*) is the change point set and probability vector that generated the data.

2.2.1 Penalized Maximum Likelihood

From the probability function written previously, it is straightforward to obtain the likelihood function

$$L(C, \mathbf{p}; \mathbf{X}) = \prod_{i=1}^{n} \prod_{j=0}^{k} \prod_{c=1+c_j}^{c_{j+1}} (1-p_j)^{\left(1-x_c^{(i)}\right)} p_j^{x_c^{(i)}} \quad .$$

Moreover, the log-likelihood is

$$l(C, \mathbf{p}; \mathbf{X}) = \sum_{i=1}^{n} \sum_{j=0}^{k} \sum_{c=1+c_j}^{c_{j+1}} \left(1 - x_c^{(i)}\right) \log(1-p_j) + x_c^{(i)} \log(p_j) \quad .$$

Fixing a set of change points $C = \{c_1, \ldots, c_k\}$, simple calculations shows that the maximum likelihood estimator for p_j is

$$\hat{p}_j = \frac{1}{n(c_{j+1} - c_j)} \sum_{i=1}^{n} \sum_{c=1+c_j}^{c_{j+1}} x_c^{(i)}, \ \ \forall \ j \in 0:k \quad ,$$

which is precisely the average of the block $c_j : c_{j+1}$.Write the maximum likelihood estimator of $\mathbf{p}$ as $\hat{\mathbf{p}}_C = (\hat{p}_1, \ldots, \hat{p}_{k+1})$, we will use the following notation

$$\hat{l}(C; X) = l((C, \hat{\mathbf{p}}_C); \mathbf{X}) \quad .$$

This way, we can focus only on the change point set when maximizing the likelihood since the estimator for p can be written conditional to C. Let $\hat{C}_k$ be the set that maximizes the likelihood among all sets of size k. That is

$$\hat{C}_k = \underset{\substack{C \subseteq 1:(m-1) \\ |C|=k}}{\operatorname{argmax}} \ \hat{l}(C; \mathbf{X}) \quad .$$

The maximum likelihood estimator can be rewritten depending only on the number of change points

$$\hat{C}_{MLE} = \underset{k \in 0:(m-1)}{\operatorname{argmax}} \hat{l}(\hat{C}_k; \mathbf{X}) \quad .$$

The number of parameters of the model is exactly the number of blocks. Since there are no restrictions on the number of blocks, the following proposition shows that the maximum likelihood estimator will always be the model that considers each point as a change point.

Proposition 2.2.1. The maximum likelihood estimator for the change point set is $1:(m-1)$.

Proof. Let $C = \{c_1, \ldots, c_k\}$ with k smaller than $m-1$, so that there exists i in $0:k$ such that $c_{i+1} \neq 1+c_i$. Define $C^* = \{c_1, \ldots c_i, c^*, c_{i+1}, \ldots, c_k\}$, where $c^* \neq c_{i+1}$. Given $\mathbf{p} = (p_1, \ldots, p_i, p_{i+1}, \ldots, p_k, p_{k+1}) \in [0,1]^{k+1}$, take $\mathbf{p}^* = (p_1, \ldots, p_i, p_i, p_{i+1}, \ldots, p_k, p_{k+1}) \in [0,1]^{k+2}$, the probability vector that repeats p_i in the block $(1+c^*) : c_{i+1}$. A straightforward calculation shows that

$$l(C, \mathbf{p}; \mathbf{X}) = l(C^*, \mathbf{p}^*; \mathbf{X}) \quad .$$

Since there are more possible values for $\mathbf{p}^*$, we have that

$$\hat{l}(C; \mathbf{X}) \leq \hat{l}(C^*; \mathbf{X}) \quad .$$

Consequently

$$\hat{l}(\hat{C}_k; \mathbf{X}) = \hat{l}(\hat{C}_k^*; \mathbf{X}) \leq \hat{l}(\hat{C}_{k+1}; \mathbf{X}) \quad .$$

□

By carefully looking at how we defined the parameter space, we can check that the model is non-identifiable. A block can be divided into two blocks with the same probability parameter. However, if we impose some regularization on the number of parameters, models with useless splits would attain higher values on the overall loss since the likelihood would be the same as the simpler model.

We will begin by considering an additive regularization that depends on the parameters.

Definition 2.2.1. Given a sample $\mathbf{X}$ and a constant $\lambda > 0$, the Penalized Likelihood is defined as

$$\mathrm{PL}(C, \mathbf{p}; \mathbf{X}) = -l(C, \mathbf{p}; \mathbf{X}) + \lambda R(C, \mathbf{p}) J(n) \quad ,$$

and the PL estimator is then

$$(\hat{C}, \hat{p})_{\mathrm{PL}} = \underset{\substack{C \subseteq 1:(m-1) \\ \mathbf{p} \in [0,1]^{|C|+1}}}{\operatorname{argmin}} \mathrm{PL}(C; \mathbf{p}; \mathbf{X}) \quad . \tag{2.1}$$

□

Notice we are considering the negative of the log-likelihood as the loss function.

The functions R and J can be of various forms, but it is common to use an information criterion like **BIC**, where $R(C, \mathbf{p}) = |C| + 1$ and $J(n) = \log(n)$. The regularization constant λ is usually fixed or estimated using cross validation.

If we make $R(C, \mathbf{p}) = R(C)$, i.e. a function that does not depend on the probability vector $\mathbf{p}$, then only $l(C, \mathbf{p}; \mathbf{X})$ depends on $\mathbf{p}$. Therefore, fixed C, the value of $\mathbf{p}$ that minimizes (2.1) will be the MLE $\hat{\mathbf{p}}_C$. In this case, we write the Penalized Maximum Likelihood estimator as

$$\hat{C}_{\mathrm{PML}} = \underset{C \subseteq 1:(m-1)}{\operatorname{argmin}} \mathrm{PL}(C; \mathbf{X}) \quad . \tag{2.2}$$

By imposing some conditions on the regularization function, we can show that the PML estimator is strongly consistent, i.e., it correctly estimates the number of change points, their position, and the block probabilities.

Definition 2.2.2. Let $C = \{c_1, c_2, \ldots, c_k\}$ be a subset of $1 : (m-1)$, $s \in 1 : (m-1)$ an integer such that $s \notin C$, and define $C' = C \cup \{s\}$, that is, C' is C with an additional division of some interval $(1 + c_j) : c_{j+1}$. A regularization function R is called piecewise increasing if $R(C) < R(C')$. Hence, every splitting we perform in an interval increases the penalization. □

The parsimony of a model is usually thought of as the number of parameters associated with the model, and we usually work with regularizations that increase with them. However, we can imbue more aspects of the problem by working with other functions.

Suppose that we are in an application in which we would like to penalize heavily concise blocks. For instance, if we had the change point sets $C_1 = \{1\}$ and $C_2 = \{\frac{m}{3}, \frac{2m}{3}\}$, we would like $R(C_2) < R(C_1)$, since the blocks in C_1 has a block of length 1. However, by choosing a regularization that is strictly increasing on the number of parameters, this is impossible.

Penalizations that are strictly increasing with $|C|$, such as the BIC and AIC, are also piecewise increasing.

We now define a property for the regularization function that is not necessary to prove the theorem, but is required to apply the computation methods provided in section 3.1.

Definition 2.2.3. A regularization function R will be called is additive if there exists a function $\rho : \mathbb{N}^2 \longrightarrow \mathbb{R}_+$ such that, for every set $C = \{c_1, c_2, \ldots, c_k\}$, we have

$$R(C) = \sum_{i=0}^{k} \rho(1 + c_i, c_{i+1}) \tag{2.3}$$

with the convention $c_0 = 0$ and $c_{k+1} = m$. □

We provide an example of a regularization that heavily penalizes concise blocks using additive regularization.

Example 2.2.1. Consider the function $\rho(r, s) = e^{m-|s-r|}$ and define the regularization as

$$R(C) = \sum_{i=0}^{k} \rho(1 + c_i, c_{i+1}) \quad ,$$

for any change point set C. This regularization captures more appropriately penalization on short blocks. Defining C_1 and C_2 as before, we have

$$R(C_1) = e^{m-1} + e^1$$

and

$$R(C_2) = 3e^{\frac{2m}{3}} \quad .$$

For $m > 6$, we have $R(C_2) < R(C_1)$, despite $|C_1| < |C_2|$.

It is easy to check that it is piecewise increasing and additive. The choice of regularization drastically changes the estimators for small sample sizes. However, the next theorem will show that, in the long run, this estimator will still converge to the true change point sets.

Theorem 2.2.2. Let $\hat{C}_{\text{PML}}$ be the estimator given by equation (2.1). Suppose that $R(C)$ is piecewise increasing and that $J(n)$ is a function that grows faster than $\log(\log(n))$ and slower than n. Suppose also that the penalization parameter λ is fixed. Then, $\hat{C}_{\text{PML}}$ is strongly consistent.

The proof of the theorem is in appendix A. If we have a fixed finite set of λ's to choose using cross-validation, the estimator is still consistent since it is consistent for each constant in the set. In practice, however, it is more common to use the data to search for the set of λ's, and hence the consistency is no longer guaranteed.

Other two considerations can be made concerning the theorem. First, notice that the regularization function **is not necessarily homogeneous on the array positions**; it only needs to be

piecewise increasing. It is useful if the indexes of the arrays have an interpretation, and distances between them can be computed and compared.

The second observation is that we might want to consider a regularization function that assigns an infinity loss if any of the blocks is too small. For instance, suppose we wished to build an additive regularization function using $\rho(r, s)$ to penalize each block, but in a way that no block is shorter than a value T. We could then define

$$\rho_T(r, s) = \begin{cases} +\infty & , \text{if} \quad |s - r| \leq T \\ \rho(r, s) & , \text{otherwise.} \end{cases}$$

and use as the new penalization function for each block. If we have that $|c_{i+1} - c_i| > T$ for all change points, then an adaptation of the proof of the theorem above will guarantee that the estimator is consistency using this type of regularization as well.

2.2.2 Cross-validation on the number of change points

Another common approach to avoid overfitting is to used resampling methods. These allow us to estimate more accurately the expected prediction error.

Cross-validation (or CV) is a resampling technique that is based on the idea of creating partitions of the data. There are several types of CVs, but we are interested in one which is more computationally feasible, the r-fold CV.

Let $X_{(i)}^{Tr}$ be the i-th train sample and $X_{(i)}^{Te}$ the i-th test sample. Consider that we have one model for each number of change points. The MLE $\hat{C}_k$ on the training sample for each set of size $k \in 0 : (m-1)$ is

$$\hat{C}_k = \underset{|C|=k}{\operatorname{argmin}} -l\left(C; X_{(i)}^{Tr}\right) \quad .$$

The prediction error on the test sample $X_{(i)}^{Te}$ is given by

$$\widehat{\mathrm{PE}}_{CVSEG}^{(i)}(k) = -l\left(\hat{C}_k, \mathbf{p}_{\hat{C}_k}; X_{(i)}^{Te}\right) \quad ,$$

that is, we are evaluating the negative log-likelihood using the parameters estimated in the training set.

We have one prediction error for each of the r splits. Let $\overline{\mathrm{PE}}_{CVSEG}(k)$ be the mean prediction error over the splits for the model with k change points, that is

$$\overline{\mathrm{PE}}_{CVSEG}(k) = \frac{1}{r} \sum_{i=1}^{r} \widehat{\mathrm{PE}}_{CVSEG}^{(i)}(k) \quad .$$

The estimated number of change points is

$$k_{cv} = \underset{k \in 0:(m-1)}{\operatorname{argmin}} \ \overline{\mathrm{PE}}_{CVSEG}(k) \quad .$$

Finally, the model is fitted again using all the data. W compute the MLE for that number of change points

$$(\hat{C}, \hat{\mathbf{p}})_{CVSEG} = \underset{\substack{|C|=k_{cv} \\ \mathbf{p} \in [0,1]^{k_{cv}+1}}}{\operatorname{argmin}} \ -l(C, \mathbf{p}; \mathbf{X}) \quad .$$

Some problems might occur in variable selection. The work of Shao [1993] showed that, for model selection in linear regression settings, some CV schemes are inconsistent, in the sense that they do not choose the correct model with probability 1 as the sample size increases. The selected set tends always to include the correct variables, but is also unnecessarily large.

2.2.3 Fused Estimation

In regression, it is usual to penalize the parameters using the ℓ_p norms, $p \geq 1$. The Lasso, created by Tibshirani [1996], uses the ℓ_1 norm to shrink the estimate of several parameters to zero. This idea is quite useful for high dimensional data where the number of parameters is much greater than the sample size.

The Fused Lasso, proposed in Tibshirani et al. [2005], is an adaptation of the Lasso to induce sparsity not only on the parameters but also on their difference. For the settings it was proposed, the independent variables' order was important, and it was expected to have blocks of parameters with the same estimate.

For our application, we consider a model similar to the Fused Lasso proposed in Tibshirani et al. [2011], in which they induce sparsity only in the difference of the parameters. We do not want to set the probability parameters to zero, only their differences to create blocks of homogeneous probabilities.

Definition 2.2.4. The Fused estimator is defined as

$$\hat{\mathbf{p}}^{\text{Fused}} = \underset{\mathbf{p}\in[0,1]^m}{\operatorname{argmin}} -\mathrm{l}(\mathbf{p};\mathbf{X}) + \lambda \sum_{i=2}^{m} |p_i - p_{i-1}| \quad , \tag{2.4}$$

where $\mathrm{l}(\mathbf{p};\mathbf{X})$ is the log likelihood considering each variable as a block. The estimated change point set will be

$$\hat{C}^{\text{Fused}} = \left\{ c \in 1:(m-1) \,|\, \hat{\mathbf{p}}_c^{\text{Fused}} \neq \hat{\mathbf{p}}_{c+1}^{\text{Fused}} \right\} \quad .$$

□

The probabilities are directly estimated; the change point set does not appear in the equation. The probabilities might also differ from the MLE. If we want the probabilities to coincide with the MLE, we estimate the change point set using the fused lasso probabilities and then recalculate them based on the set.

The penalization constant can again be chosen using cross-validation. Notice we incorporated the penalization function of n in λ as it is usual for Lasso methods.

Chapter 3

Computational Aspects and Simulation

In this chapter, we begin discussing the computational techniques used to compute the estimators. Under some conditions on the penalization function, it is possible to obtain the exact solution for all estimators using dynamic programming. A greed solution using hierarchical segmentation is also presented.

Then, we analyze the results of an extensive simulation study to compare the convergence of the methods and answer some raised questions. All simulations were made using **bincpd**, our R package created to implement the methods here studied. An example of usage of the package is given in Appendix C.

3.1 Computational Aspects

In Chapter 2 we observed that, given the change points set $C = \{c_1, \dots c_k\}$, the MLE $\hat{p}_C$ is the probability vector where each entry is the block mean. Given an integer interval $r : s$, we define the negative block entropy as

$$Q(r : s; \mathbf{X}) = n|s - r|(\hat{p}_{r:s} log(\hat{p}_{r:s}) + (1 - \hat{p}_{r:s}) log(1 - \hat{p}_{r:s})) \quad ,$$

where $\hat{p}_{r:s}$ is the block mean. In appendix A, we show that

$$\hat{l}(C; \mathbf{X}) = l((C, \hat{\mathbf{p}}_C); \mathbf{X}) = \sum_{i=0}^{k} Q((c_j + 1) : c_{j+1}; \mathbf{X}) \quad .$$

Remember that the PML estimator is given by

$$(\hat{C}, \hat{\mathbf{p}}_C) = \underset{C \subseteq (1:(m-1))}{\operatorname{argmin}} \; -\hat{l}(C; \mathbf{X}) + \lambda R(C) J(n) \quad .$$

As discussed before, our only task is to find the change point set C since the estimator for $\mathbf{p}$ is $\hat{\mathbf{p}}_C$. However, there might be too many points to evaluate since there are 2^{m-1} possible change point sets. If we make no assumptions on the regularization function R, the exact solution might require us to check every set. To avoid this, we impose some restrictions on R to search for the exact solution computationally feasible. If we want an even faster solution, algorithms with greed heuristics can be used to obtain an approximate solution.

3.1.1 Dynamic Programming

Dynamic programming is a computer science paradigm used to obtain exact solutions to problems with a recursive structure. By storing information and checking them only when needed, we avoid unnecessary repeated calculations, which usually is a massive problem in naive recursive algorithms. It can be applied in a wide range of problems, including optimization, and it reduces the complexity considerably compared to a simple recursion or brute force optimization.

Let us show that, by imposing the regularization function to be additive, dynamic programming can be used to calculate the estimators. Suppose that R is additive and write

$$R(C) = \sum_{i=0}^{k} \rho(c_i + 1, c_{i+1}) \quad .$$

The optimization function becomes

$$\begin{aligned} -\hat{l}(C; \mathbf{X}) + \lambda J(n) R(C) &= \sum_{i=0}^{k} -Q((c_j + 1) : c_{j+1}; \mathbf{X}) + \lambda J(n) \rho(c_i + 1, c_{i+1}) \\ &= \sum_{i=0}^{k} \tilde{Q}((c_j + 1) : c_{j+1}; \mathbf{X}) \quad , \end{aligned}$$

where

$$\tilde{Q}((c_j + 1) : c_{j+1}; \mathbf{X}) = -Q((c_j + 1) : c_{j+1}; \mathbf{X}) + \lambda J(n) \rho(c_i + 1, c_{i+1}) \quad .$$

This equation shows that we can completely decouple the contributions from different blocks. If we define

$$F_k(i) = \min_{C \in \mathcal{P}(1:i), |C|=k} \left\{ \sum_{i=0}^{k} \tilde{Q}((c_j + 1) : c_{j+1}; \mathbf{X}) \right\} \quad ,$$

the best estimator of k blocks would attain a regularized value of $F_k(m)$. If we could find each of these, our task would only be to compare them. But notice that

$$F_k(i) = \min_{c \in (k-1):(i-1)} \{F_{k-1}(c) + \tilde{Q}((c + 1) : i; \mathbf{X})\} \quad ,$$

which establishes a recursion equation for the values of $F_k(i)$, with k varying in 1 to m and i from k to m. The values of $F_1(i)$ can be directly computed, and then we use the recursion to compute values until $F_m(m)$. In each step, we store each of the compute $F_k(i)$, hence using dynamic programming to avoid repetitive calculations.

In applications, it is also useful to restrain the number of change points to a maximum of $K_{max} \leq m - 1$. The pseudocode for the implementation of the dynamic programming segmentation algorithm is provided below. We will omit $\mathbf{X}$ from $\tilde{Q}$ to simplify the notation, writing $\tilde{Q}(r : s)$ instead of $\tilde{Q}(r : s; \mathbf{X})$.

Algorithm 1 Dynamic Programming Segmentation Algorithm

procedure DPS(K_{max})
  $regLossMat \leftarrow array[0, \ldots, K_{max}][1, \ldots, m]$ ▷ Initializing DP data structure to compute F
  $cpMat \leftarrow array[1, \ldots, K_{max}][1, \ldots, m]$ ▷ data structure to compute change points
  $regLossMin \leftarrow \infty$ ▷ Global minimum
  $\hat{k} \leftarrow None$
  for $k = 0, \ldots, K_{max}$ **do**
    for $i = k, \ldots, m$ **do**
      if $k = 0$ **then** ▷ Base cases when there are no change points
        $regLossMat[0, i] \leftarrow \tilde{Q}(1 : i)$
        continue ▷ Skip the rest of the iteration
      end if
      $regLoss \leftarrow \infty$ ▷ regularized loss at current iteration
      for $c = (k-1), \ldots, (i-1)$ **do**
        if $regLoss > regLossMat[k-1, c] + \tilde{Q}((1+c) : i)$ **then**
          $regLoss \leftarrow regLossMat[k-1, c] + \tilde{Q}((1+c) : i)$
          $bestCP \leftarrow c$
        end if
      end for
      $regLossMat[k, i] \leftarrow regLoss$
      $cpMat[k, i] \leftarrow bestCP$ ▷ Location of the best change point
    end for
    if $regLossMin > regLossMat[k, m]$ **then** ▷ Update best number of change points
      $regLossMin \leftarrow regLossMat[k, m]$
      $\hat{k} \leftarrow k$
    end if
  end for
  $\hat{C} \leftarrow \emptyset$
  $lastCP \leftarrow m$
  for $k = \hat{k}, \ldots, 1$ **do** ▷ Retrieve change point set
    $lastCP \leftarrow cpMat[k, lastCP]$
    $\hat{C} \leftarrow \hat{C} \cup \{lastCP\}$
  end for
  return $\hat{k}, \hat{C}$
end procedure

The algorithm computes the exact solution, has storage complexity of $O(K_{max}m + m^2)$ and time complexity of $O(nm + m^2 + K_{max}^2 m)$, K_{max} being the maximum number of change points allowed. The $O(nm)$ and $O(m^2)$ comes from the fact that we have to scan all data to build a $O(m^2)$ matrix to evaluate $\tilde{Q}$. If we set $K_{max} \in O(m)$, the algorithm is $O(m^3)$, and hence is very slow for big values of m.

The data structures used in the algorithm are triangular matrices since we do not need the entries below the principal diagonal. Storage space can be saved using an array of lists instead, but we preferred this implementation for simplicity.

The $cpMat[k, i]$ entry represents the best last change point of the split with k change points prior to i. We can avoid using this matrix and recompute the change point set directly at the end, trading storage space for time complexity. However, this matrix is useful for the $CVSEG$ estimator since we need the change point set for every value of k.

3.1.2 Hierarchical Segmentation Algorithm

Albeit the dynamic programming algorithm is polynomial in run time and storage, it still can be relatively slow, especially if m is large. For faster computation, we can give up on the exact solution and obtain an approximation using the hierarchical segmentation algorithm, also known as binary segmentation algorithm, one of the most studied algorithms in change point detection.

The idea is to search for the best splitting point in a given interval and then call the algorithm recursively in each of the smaller intervals until no more change points are detected.

Let $I = r : s$ be an integer interval, define

$$h(I; \mathbf{X}) = \underset{c \in r:(s-1)}{\operatorname{argmin}} \left(\tilde{Q}(r : c; \mathbf{X}) + \tilde{Q}((c+1) : s; \mathbf{X}) \right) \quad ,$$

with the convention $\tilde{Q}(\emptyset; \mathbf{X}) = 0$. Hence h is a function that receives an interval I and returns the best splitting point. By running this procedure recursively in each par of splitted intervals, we have a greed solution for the optimization problem.

The pseudocode for the hierarchical segmentation algorithm is provided below.

Algorithm 2 Hierarchical Algorithm

```
global variables
    Ĉ ← ∅                                          ▷ Empty change point set.
end global variables
procedure HS(leftIndex, rightIndex)
    regLossMin ← Q̃(leftIndex : rightIndex)          ▷ Loss value when there is no split.
    splitIndex ← None
    for c = leftIndex, ..., (rightIndex − 1) do
        regLoss ← Q̃(leftIndex : c) + Q̃((c + 1) : rightIndex)
        if regLoss < regLossMin then
            regLossMin ← regLoss
            splitIndex ← c
        end if
    end for
    if splitIndex ≠ None then                      ▷ Recursive calls if there was a split!
        Ĉ ← Ĉ ∪ {splitIndex}                       ▷ Update change point set
        HS(leftIndex, splitIndex)
        HS(splitIndex + 1, rightIndex)
    end if
end procedure
```

The desired change point set is obtained by running $HS(1, m)$. Notice we have to compute $\tilde{Q}$, and we either store all possible values in the same fashion of the dynamic programming or evaluate them on the fly, only using the pre-computed sufficient statistics.

In the worst-case scenario, the algorithm has a run time of order $O(m^2)$. As we will prove later, the algorithm will be asymptotically correct, so it will almost surely do $2|C| + 1$ recursive calls: $|C|$ calls to obtain each change points and $|C| + 1$ to check there are no points between change points. Considering both splitted intervals on each recursion step, we have $O(m)$ comparisons and $O(m)$ computations of $\tilde{Q}$ to make. We also have to compute beforehand the sufficient statistics p, the empiric probability of each row, and N, the vector of column sample sizes. To compute them, we need $O(nm)$ time complexity and $O(m)$ storage space. Hence, the algorithm will run on $O(m(n + |C|))$ time complexity.

In Chapter 2 we stated that the PML estimator, obtained exactly by the dynamic programming algorithm, is consistent. The following theorem guarantees that, even by using the approximate solution, we still have consistency.

Theorem 3.1.1. Let $\hat{C}_{\text{HS}}$ estimator given by the hierarchical segmentation algorithm. Suppose that $R(C)$ is piecewise increasing and that $J(n)$ is a function that grows faster than $\log(\log(n))$ and slower than n. Suppose also the penalization parameter λ is fixed. Then, $\hat{C}_{\text{HS}}$ is strongly consistent.

The proof of the theorem is in appendix A.

3.1.3 Fused Computation

The solution for 2.4 can be obtained using convex optimization. It is show in Kim et al. [2009] how to transform the Fused Lasso optimization problem to a Lagrange dual and how to solve it. They also show that the problem can be extended to other convex loss functions such as entropy-like losses.

It is easy to show that the regularized loss satisfies the properties of disciplined convex programming. For that purpose, we used the R package CVXR to solve the disciplined convex optimization problem.

3.2 Metrics

Before we present the simulations, we first discuss common metrics used in change point detection.

There are several choices for evaluation metrics to study the convergence of the estimated change point set, as discussed in Truong et al. [2020]. We present three of them here: the Hausdorff distance, the symmetric difference metric, and the Rand Index.

Let C_1 and C_2 be subsets of $1:(m-1)$ The Hausdorff distance between the sets is defined as

$$d_H(C_1, C_2) = \max\left(\max_{c_1 \in C_1} \min_{c_2 \in C_2} |c_1 - c_2|, \max_{c_2 \in C_2} \min_{c_1 \in C_1} |c_1 - c_2|\right) \quad .$$

The Hausdorff Distance is indeed a metric for finite sets and is a common metric in applied statistics.

The symmetric difference metric is defined as

$$d_{sd}(C_1, C_2) = |C_1 \triangle C_2| \quad ,$$

where $C_1 \triangle C_2 = (A \cup B) \backslash (A \cap B)$. Instead of calculating the distance between the elements of each set, the symmetric difference just counts how many elements belong exclusively to one of the sets.

The Rand Index is a measure of similarity between sets or partitions, and it is commonly used in cluster analysis. Let $A = \{A_i\}_{i=1}^{r}$ and $B = \{B_j\}_{j=1}^{s}$ be partitions of $1:m$, the Rand Index is defined as

$$R = \frac{a+b}{\binom{m}{2}} \quad ,$$

where a is the number of pairs $(c_1, c_2) \in (1:(m))^2$, $c_1 \neq c_2$, that both elements belong to a single set in partition A and a single set in partition B, and b is the number of pairs that these two elements belong to different sets in A and different sets in B. The term $a+b$ is measuring the agreements between the clusterings: a is counting how many pairs appear together in both partitions, and b is counting how many appear separated in both partitions. Then, we divide by the total number of pairings, which is $\binom{m}{2}$, so that the index is always in $[0,1]$. A value of 0 means total disagreement, and a value of 1 means a perfect match, i.e., both clusterings are identical.

Given a change point set $C = \{c_1, c_2, \dots c_r\}$, the partition associated would be $\{C_1, \dots, C_{r+1}\}$, where $C_i = (c_{i-1}+1):c_i$, using the convention $c_0 = 0$ and $c_{r+1} = m$. These are exactly the blocks obtained after the segmentation.

In order to compute the Rand Index, we have to build a contingency table for the intersections of the sets of the partitions and, by doing so, we have to check each element of $1:m$. However, since the change points are sufficient to describe the partitions, we could expect to have an equation for the Rand Index that only depends on the change points, so that it is not necessary to check each point.

Proposition 3.2.1. Let $C = \{c_1, c_2, \dots c_r\}$ and $C^* = \{c_1^*, c_2^*, \dots c_s^*\}$ be change point sets whose partitions are those described as above. Define $c_0 = c_0^* = 0$ and $c_{r+1} = c_{s+1}^* = m$. Then, the Rand Index is given by

$$R = 1 - \frac{\sum_{i=0}^{r}\sum_{j=0}^{s} M_{ij}|c_i - c_j^*|}{\binom{m}{2}} \quad ,$$

where

$$M_{ij} = \max\left(0, \min\left(c_{i+1}, c_{j+1}^*\right) - \max\left(c_i, c_j^*\right)\right) \quad .$$

The proposition is proved in Appendix B. It provides an algorithm to compute the Rand Index in $O(rs)$ time complexity and $O(1)$ in storage usage for change point detection problems. The original algorithm is $O(m + rs)$ in time and $O(rs)$ in storage since it needs to go through $1:m$ to build a $r \times s$ contingency table.

3.3 Simulations

In this section, we discuss a simulation study that attempts to gain some insight on the following questions:

(1) **If we fix the block lengths, number of change points, and probabilities, does increasing m makes the models converge faster or slower?**

The speed of convergence depends on several parameters: the number of variables m, the blocks probabilities and their successive differences, the proportional size of each block ($|I|/m$), and the number of change points k. If we fix all the parameters and only allow m to vary, would increasing it enhance convergence? On the one hand, we have more samples for each block, improving the estimation of each blocks probability parameter, and hence making it easier to discriminate between the blocks. On the other hand, we have more possible models and comparisons to make, which could slow down convergence.

(2) **How well do different metrics discriminate between the performance of the models?**

Any metric defined on the change point sets can be used to study convergence. However, are all of them equally good at evaluating models for change point detection problems, in the sense that they give very different scores to very different estimates?

(3) **Does the PML fit using dynamic programming converge much faster than if we use hierarchical segmentation?**

We know that dynamic programming gives us the exact answer and that hierarchical algorithm gives only an approximate solution. However, does this imply that the convergence speed is worth the extra computation?

(4) **How does model performances compare to each other?**

We have proved that the **PML** model is consistent using both the DP and HS algorithms, but we do not know if this property holds for the other models. Can the simulation studies give us some insight if the model is not consistent? Moreover, does consistency imply that the **PML** will in general present better results than other models?

There are other questions that would be interesting to tackle. The first one would is: how does the change points proportion k/m affects the models' convergence? The way this simulation study was performed can not provide good insight to this question since it is hard to fix the other parameters in this case.

The second question is on how the probability p_i affect the convergence? Values closer to 0 or 1 should be easier to estimate, and values closer to $\frac{1}{2}$ would be harder. Besides that, the gap of probabilities between blocks, that is $|p_{c_i} - p_{c_{i+1}}|$ would also influence on the convergence of the model. Since there is no straightforward way to answer this question together with the chosen questions, we do not analyze this question.

3.3.1 Simulation settings

Data sets and parameters

The number of change points k was varied between 10, 50 and 90. The vector size m was also chosen in a set of three possible values: 100, 200 and 500. This amounts to a total of 9 scenarios.

Each entry of the probability vector was sampled from a uniform distribution, independently from others, **for each different number of the change points**. For instance, this means that for $k = 10$, the probability vector was exactly the same, not varying as m changes from 100 to 200 or 500.

For the change points, **for each different number of the change points**, we sampled then uniformly when $m = 100$, and re-scaled them for $m = 200$ and $m = 500$ in order to maintain the block size proportions.

We considered sample sizes ranging from 50 to 1000 samples in steps of 50. Given the change points and probabilities, for each sample size, we simulated 100 data sets and computed the estimators.

This setting of the parameters allows us to answer more appropriately question **(3)**.

Models

The following models were fit on the simulated data

Model	Algorithm	λ	$J(n)$	$R(s:t)$
PML	DPS	1	$log(n)$	1
PML	DPS	1	$\sqrt{n}$	1
PML	HS	1	$log(n)$	1
CVSEG	DPS	-	$log(n)$	1
CVPML	DPS	{0.1, 1, 10}	$log(n)$	1

Table 3.1: *Models considered for the simulation study. Both dynamic segmentation (DPS) and hierarchical segmentation (HS) were considered for comparison. For $J(n)$, we chose a more soft penalization $log(n)$, and a more aggressive penalization $\sqrt{n}$. The regularization $R(s:t)$ is the penalization applied to the integer interval $s:t$.*

.

Evaluation

We will provide graphical information on the Hausdorff, symmetric difference, and Rand Index metrics using boxplots, along with the estimated number of change points.

3.3.2 Results

We finish discussing each question proposed, displaying selected Figures that provide the most insight. All graphics are provided in appendix E.

Increasing m

Increasing the number of rows m while maintaining other parameters has two contrasting aspects: we increase the number of sample sizes inside each block, improving our estimation on each probability parameter, but have a larger number of models to compare. Figures 3.1 and 3.2 shows the difference in convergence of the CVSEG and DPS estimators as m varies.

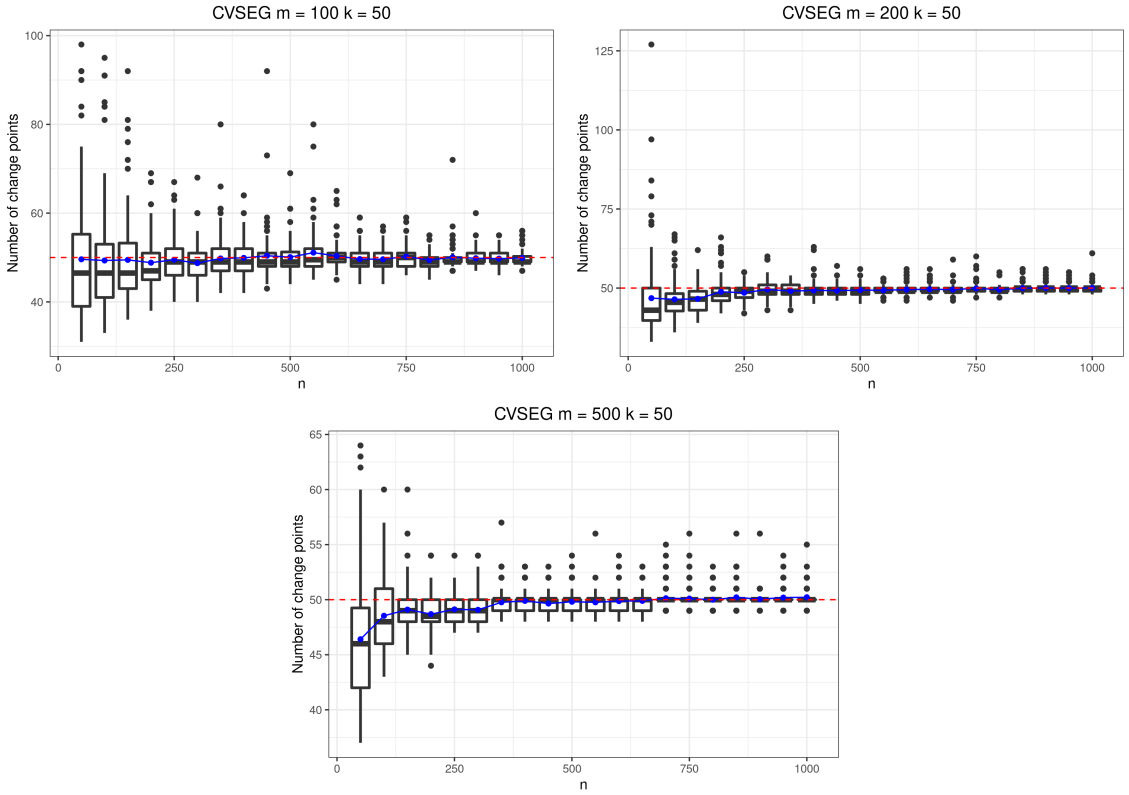


Figure 3.1: *Number of change points estimated by the CVSEG algorithm for different values of m and $k = 50$. The blue line shows the mean number of change points detected for that sample size.*

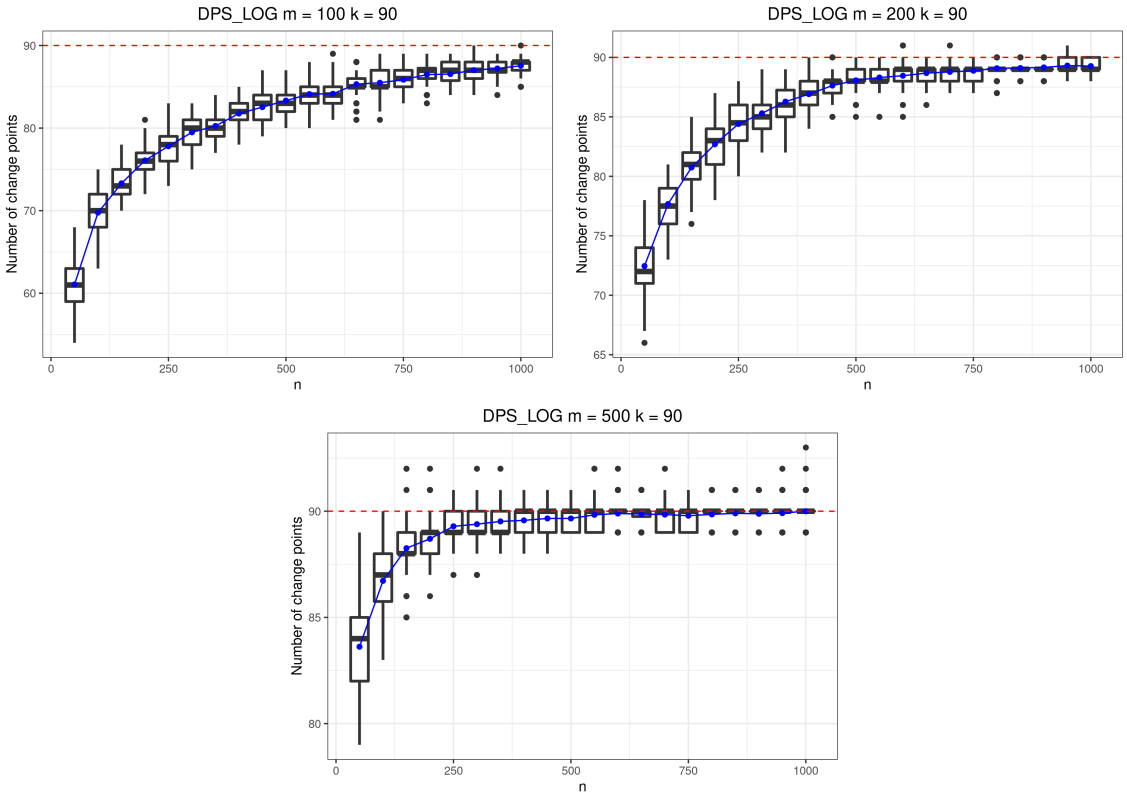


Figure 3.2: *Number of change points estimated by the DPS algorithm for different values of m, with $J(n) = log(n)$ and $k = 90$. The blue line shows the mean number of change points detected for that sample size.*

We observe that the number of change points is estimated more accurately as m grows for both models. As sample sizes grow, the estimates converge faster to the true number of change points. It might seem that the variance has increased for both methods, but this could be due to graphic re-scaling, as we compare variances in Figures 3.3.

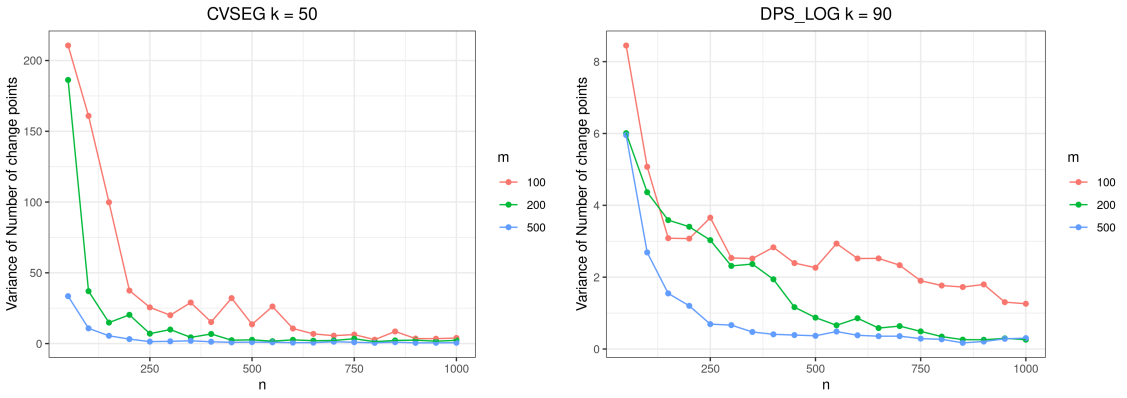


Figure 3.3: *Variance of the number of change points estimated by the methods above as sample size increases.*

The initial underestimation of the DPS and HS methods seems to be less heavy as m increases. This can be seem quite clearly looking at the plots of the DPS algorithm with $J(n) = \sqrt{n}$ and $k = 90$. Even the case with most samples ($n = 1000$) for $m = 100$ has worst estimates than the case with smaller samples ($n = 50$) for $m = 500$.

It is also important to notice that the penalizations considered do not vary with m directly. If this were the case, for instance, multiplying an extra $J(m)$, this effect might not be observed.

Comparing Metrics

Hausdorff When analyzing the performance of the models using graphical analysis, the Hausdorff distance plot shows us an interesting phenomenon. For some models and sample sizes, the convergence seems "hard stuck" because the performance does not seem to improve with the sample size. However, if we use other metrics for the analysis, we can see that the model is performs better. Figure 3.4 shows an example of this scenario for the DPS estimator with $J(n) = \sqrt{n}$.

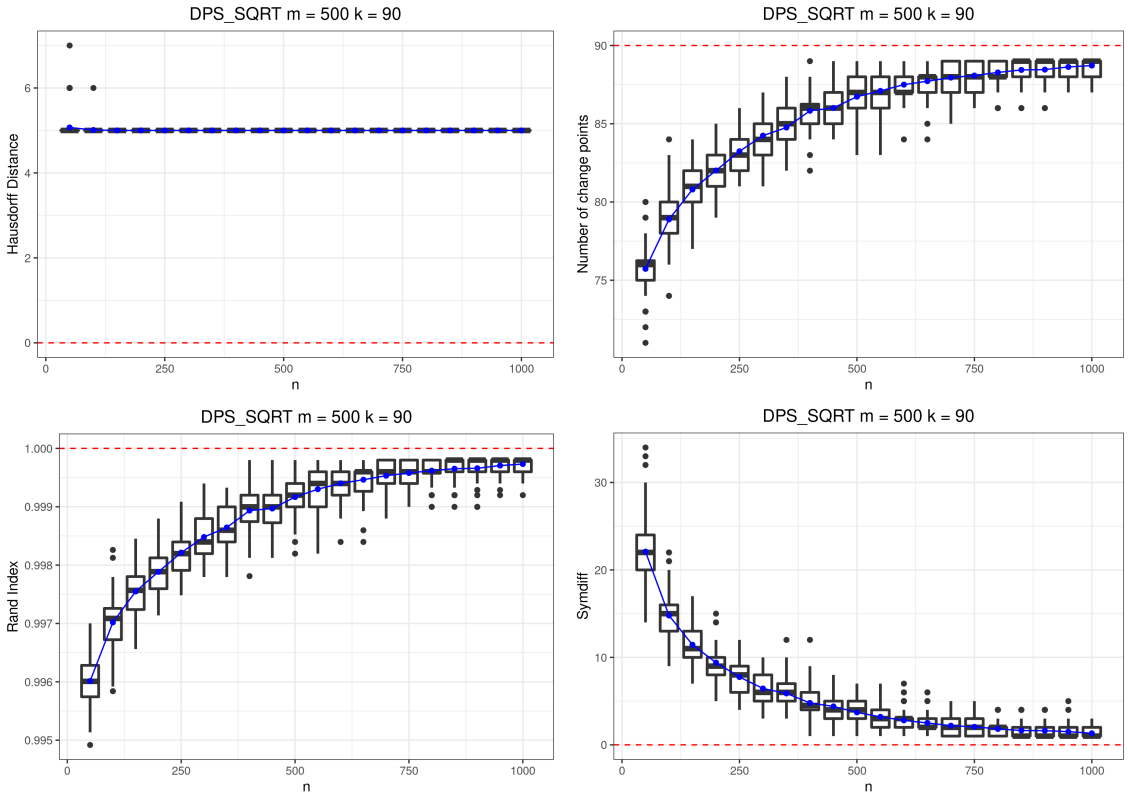


Figure 3.4: *Number of change points estimated, Hausdorff distance and Symmetric difference for the DPS with* $J(n) = \sqrt{n}$ $m = 500$ *and* $k = 90$

Notice that the model initially underestimates the number of change points, but is yielding better results as the sample size n grows, as shown on both the number of change points estimated and the symmetric difference graphic. However, we can not see this effect for the Hausdorff distance, as the value of the metric collapses on a single value for sample sizes greater than 100.

From the perspective of how the metric is calculated, this should not be a surprise since its value depends only on the distance between change points and not the actual accuracy. For instance, in a setting that every odd point is a change point, a model that estimates every even point as a change point will achieve a Hausdorff distance of 1. That is, a model that misses every change point achieves the lowest positive possible value of the metric.

To show a simulation scenario similar to what was described, consider the results in Figure 3.5 for the DPs with $J(n) = \log(n)$ and $m = 100$.

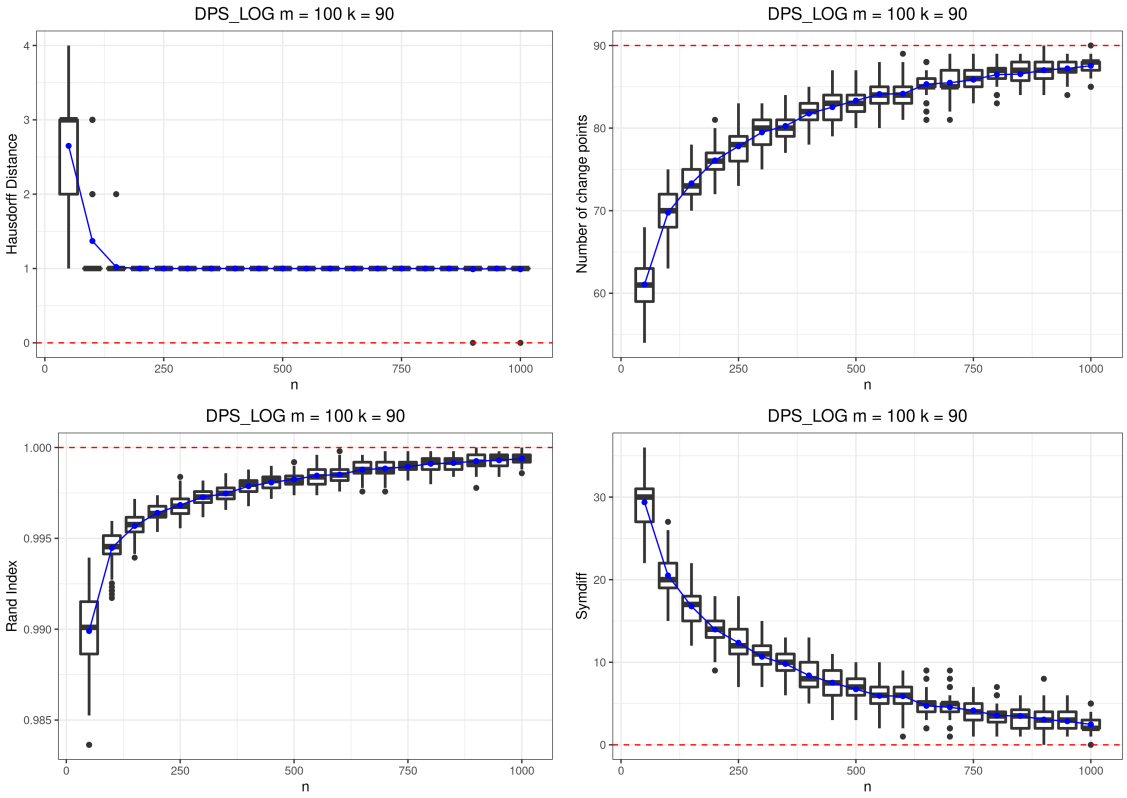


Figure 3.5: *Number of change points estimated, Hausdorff distance and Symmetric difference for the DPS with* $J(n) = \log n$ $m = 100$ *and* $k = 90$

For $n > 100$, the Hausdorff distance collapses on 1, which, if we blindly rely on this metric, will think that the estimate is very close to the ground truth. However, by looking at the other metrics, especially the number of change points, we see that the model is nowhere near the ground truth at $n = 100$ and is still converging.

At first glance, this seems to be a problem, but it is context-dependent. For many change point problems, we are not interested in estimating the precise location of the changes but rather estimating them within a $O_p(1)$ error bound. Indeed, the review by Truong et al. [2020] defines consistency as asymptotically estimating correctly not the change points c_i themselves, but rather the change point fractions $\frac{c_i}{m}$.

Rand Index Another interesting pattern seen in both figures is the Rand Index scale. We can see that the metric is varying, and the model is converging, but it is greater than 0.99, a value very close to 1. On the contrary, the symmetric distance and number of change points are varying considerably.

When analyzing the convergence, using just one metric to measure the convergence might not be the best approach.

HS vs DPS

Since the dynamic programming algorithm is exact, we would expect it to display better results than the approximate estimator yielded by the HS. However, the comparison in the simulation suggests an unexpected result. In most cases, the HS seems to have the same performance, if not better, than the DP.

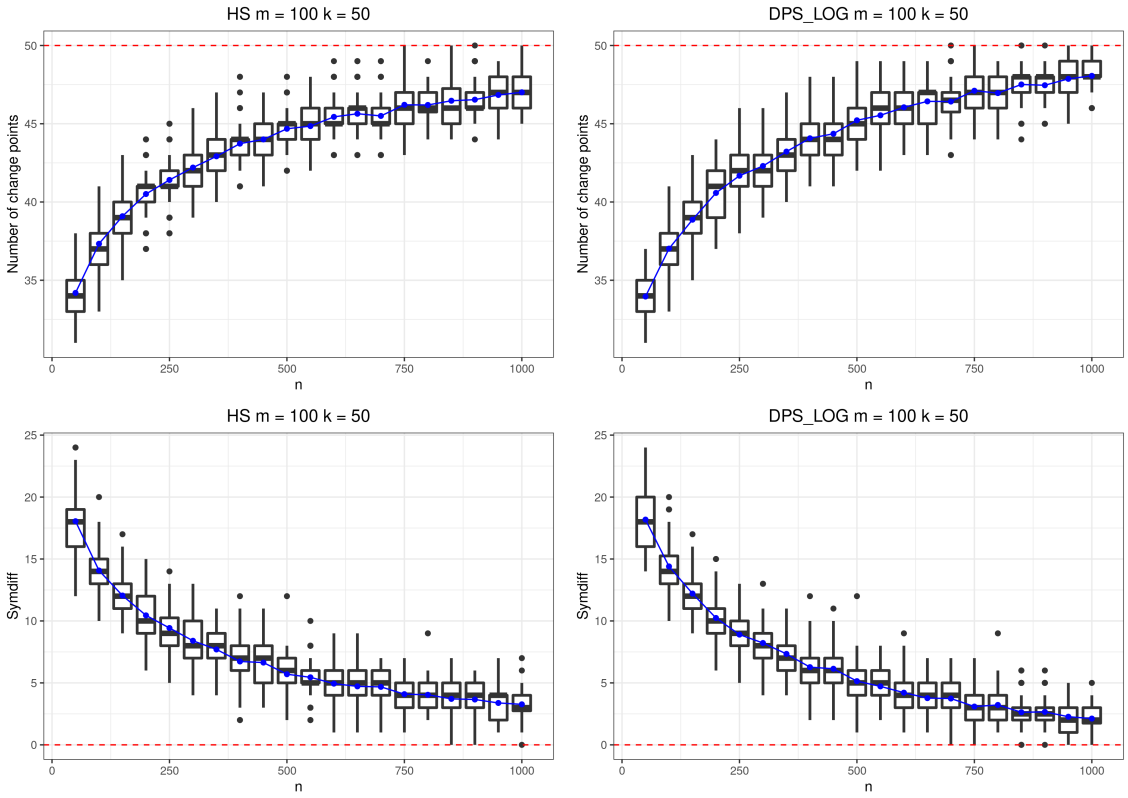


Figure 3.6: *Number of change points estimated and Symmetric difference for the HS and DPS algorithms for* $m = 100$ *and* $k = 50$*. The HS algorithm seems to perform better in this scenario.*

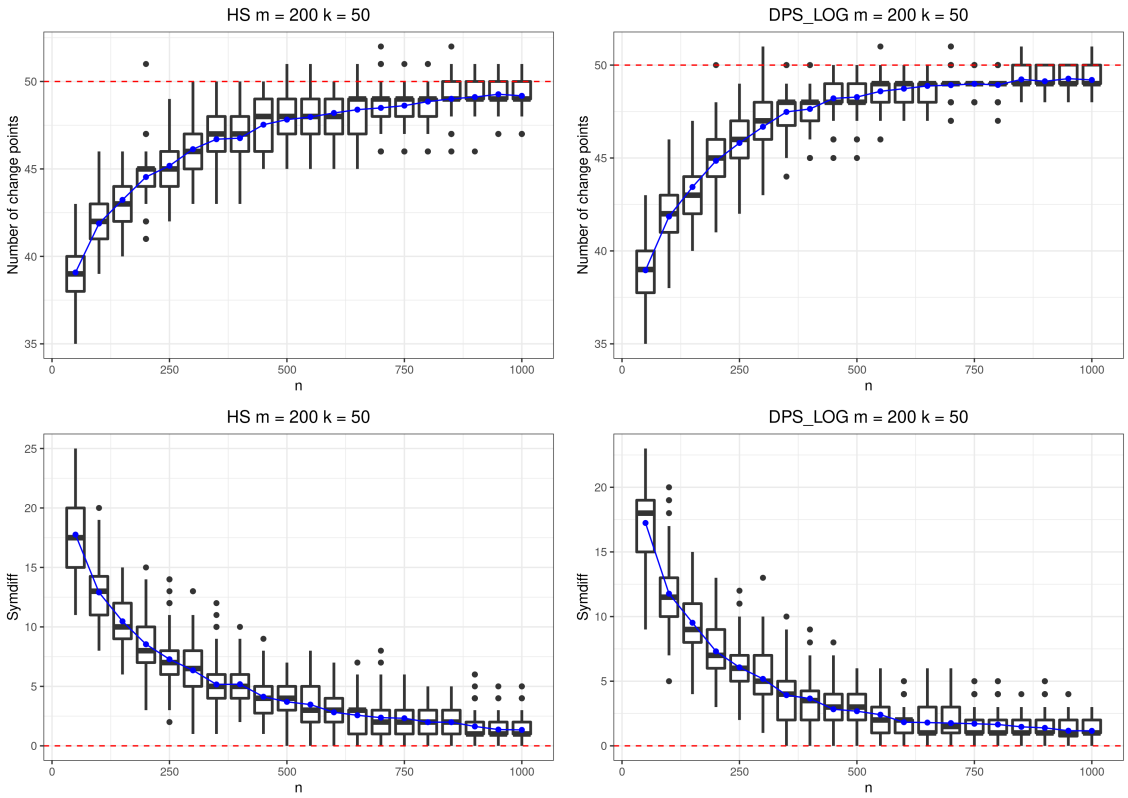


Figure 3.7: *Number of change points estimated and Symmetric difference for the HS and DPS algorithms for* $m = 200$ *and* $k = 90$*. The algorithms display similar results.*

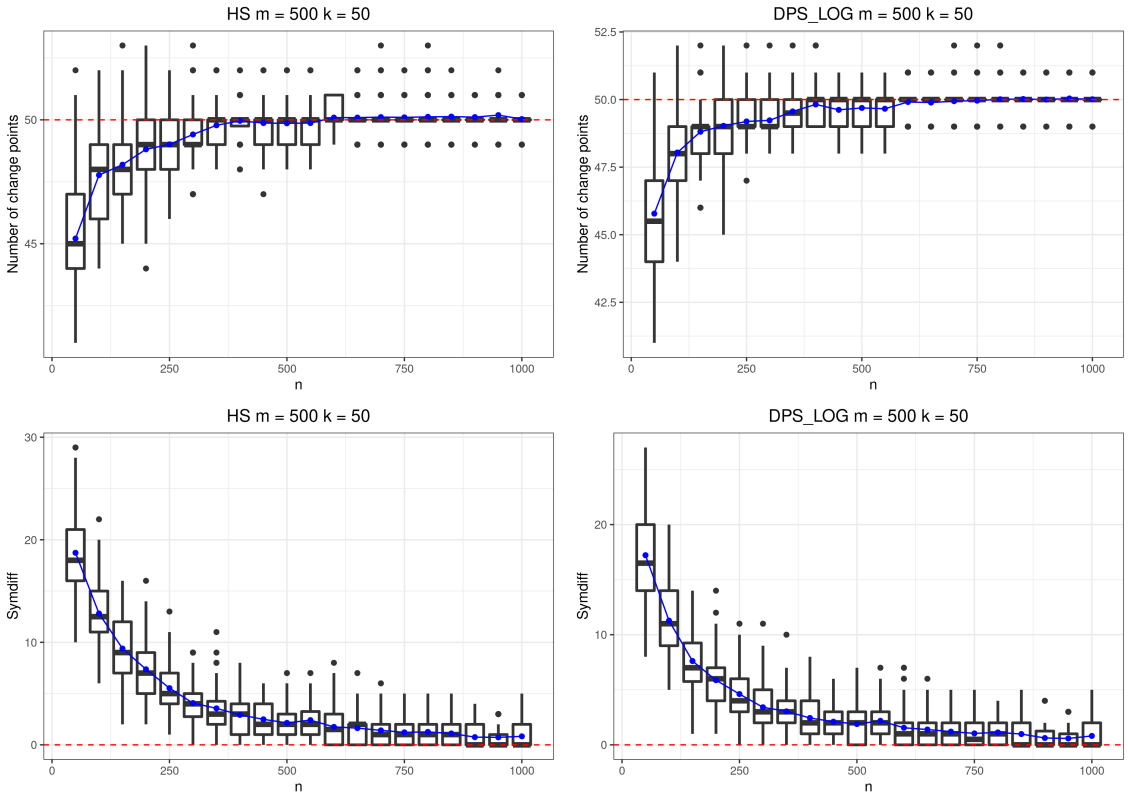


Figure 3.8: *Number of change points estimated and Symmetric difference for the HS and DPS algorithms for* $m = 500$ *and* $k = 90$*. The algorithms display similar results.*

Figures 3.7 and 3.8 shows scenarios where they had equivalent performance, and 3.6 one that HS was better. This results advances the idea that the greed solution to this particular problem might be as good as the exact one in practice.

Performance comparison

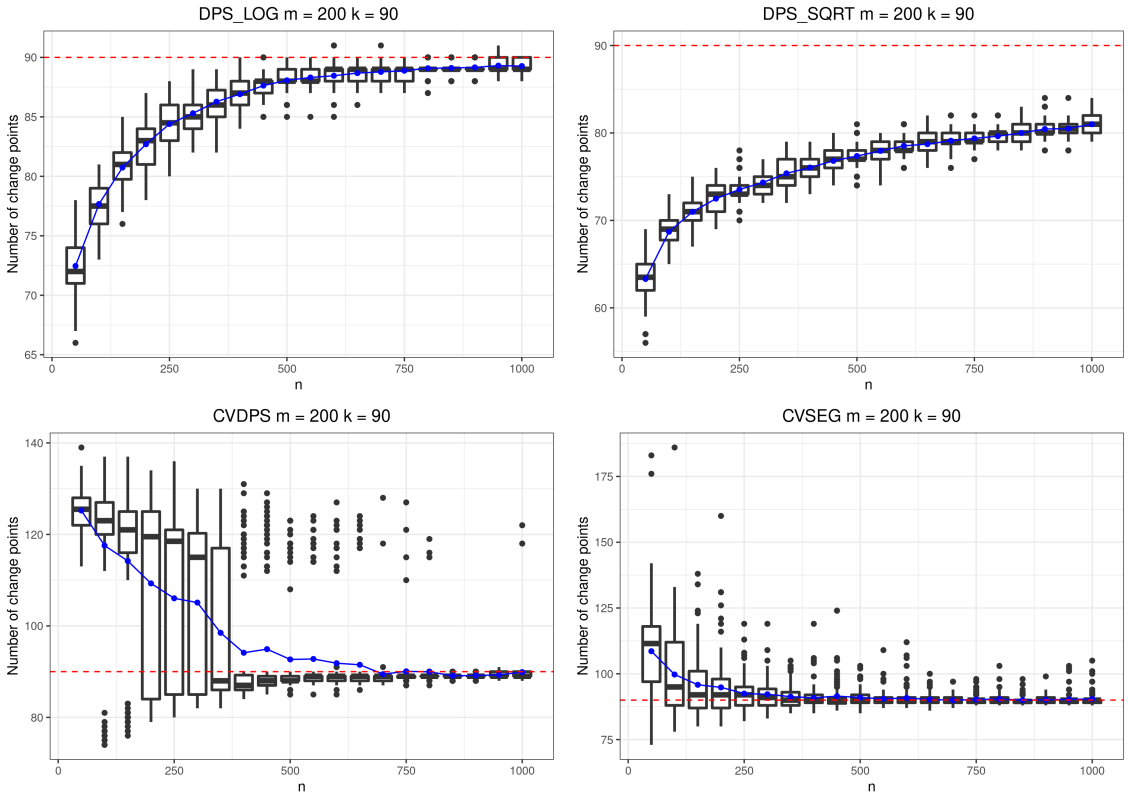


Figure 3.9: *Number of change points estimated on the* $m = 100$, $k = 90$ *scenario by the CVSEG, DPS with* $J(n) = \sqrt{n}$ *and* $J(n) = log(n)$ *and CVDPS with* $J(n) = log(n)$

The DPS and HS models underestimated the number of change points for almost all values of k and m. It is important to remember that we fixed the penalization constant λ as 1. If we chose smaller values, the models could overestimate the number of change points.

In Chapter, 2 we stated a theorem that guarantees the consistency of our method independently of the choice of the penalization constant. However, it does not guarantee a good performance. For the $m = 200$ and $k = 90$ scenario, the models heavily underestimate the number of change points, especially for smaller sample sizes. Since the box plot width is small, it seems that the models are heavily biased.

A popular strategy to chose λ is to use the computer-intensive k-fold cross-validation. We can see in the simulation results that including 0.1 in the penalization constant aids against underestimation, but could lead to overestimation, as seen in Figure 3.9. Therefore, choosing the constant set ad hoc leads a similar problem.

The choice of the penalization function $J(n)$ is also very important. In a scenario where the model already underestimate the number of change points with $J(n) = \log(n)$, choosing a stronger penalization such as $\sqrt{n}$ only worsens the convergence. A common strategy to avoid choosing $J(n)$ is to incorporate it into $\lambda = \lambda(n)$ and estimate the hyperparameter.

Chapter 4

Application

We introduce the problem of detecting islands of long runs of homozygosity (ROH) on SNP data from a population of unrelated individuals. The necessary concepts in genetics are presented in order to understand the purpose of the scientific study.

Then, we pose the problem as an offline binary change point detection problem. We apply the methods presented in this work and compare the results with state of the art methods applied in genetics.

4.1 Genetics concepts and problem formulation

4.1.1 ROH and ROH islands

In diploid organisms, such as humans, each individual's genome is organized into pairs of chromosomes, each half inherited from each parent. When an individual is an offspring of biologically related parents, both chromosomes of the same pair can share identical segments, creating long stretches of consecutive homozygosity, known as runs of homozygosity (ROH).

In the last decades, studies on the identification of ROH carried out in human populations have revealed the presence of ROH even in cosmopolitan non-inbred populations, disclosing an increment of inbreeding levels and the consequent reduction of genetic diversity of populations, which is proportional to the walking distance from Africa, as expected by the out-of-Africa model of human colonization Ceballos et al. [2018], Kirin et al. [2010], Lemes et al. [2018], Leutenegger et al. [2011], Pemberton et al. [2012].

The distribution of ROH along the chromosomes is very uneven, resulting in some genomic regions having significant absence (coldspots) or excess of ROH (ROH islands) Ceballos et al. [2018]. The mechanisms for the emergence of these regions are still under discussion. For example, there is evidence that ROH islands could represent regions that harbor genes target of positive selection since low-recombination regions commonly are locations of selective sweeps, in which a new beneficial mutation increases in frequency and becomes fixed, causing the overall reduction in genetic diversity of the region Ceballos et al. [2018], Pemberton et al. [2012].

4.1.2 SNPs

The human genome has 23 pairs of chromosomes with more than 3 billion base pairs. These codify the instructions that define and maintain the individuals. However, when comparing two humans at random, more than 99.9% of these base pairs are almost identical. If the aim is to study variability in human traits, it is more interesting to study regions of the chromosomes that have significant variance in human populations. Single Nucleotide Polymorphisms, or SNPs, are special biological markers consisting of base pairs with a sufficiently large variation in the population. The threshold might vary, but it is usual to consider a variation of at least 1%.

An SNP array is just a sequence of SNPs as they appear in the chromosome, preserving the order. For instance, for the chromosome pair 22, the second smallest human chromosome, there are

around 49 million base pairs, but the number of SNPs is usually much smaller, of the order of 10^3 or 10^4, depending on the threshold selected. Working with SNPs drastically reduces the number of variables, but still preserves most of the variability observed.

4.1.3 Framing the problem

The scientific endeavor is to understand the role of the ROH islands. However, in order to study them, we first need to know their locations. Hence, the statistical problem at hand is, given a sample of SNP arrays from a population of unrelated individuals, to detect ROH islands' regions.

How do we translate these genetic concepts into mathematical ones which we can work with? First, each individual sample is an array of length m, where m represents the total number of SNPs in that chromosome. Each entry of this array is either 0 or 1; 0 coding for heterozygote, 1 coding for homozygote.

ROHs are "long" sequences of homozygote SNPs; hence, they are "long" sequences of 1's in the codified array. The actual size of "long" varies a lot, but the idea is that it is enough to be unlikely to arise by chance, but rather by a biological mechanism. Finally, ROH islands are regions in which ROH are most frequent in that population. That is, the positions in the array with a higher frequency of "long" sequences of 1's passing through them.

Figure 4.1 shows an oversimplified representation of the problem. It is not a precise representation of the real data, but rather a schematization to understand the concepts discussed.

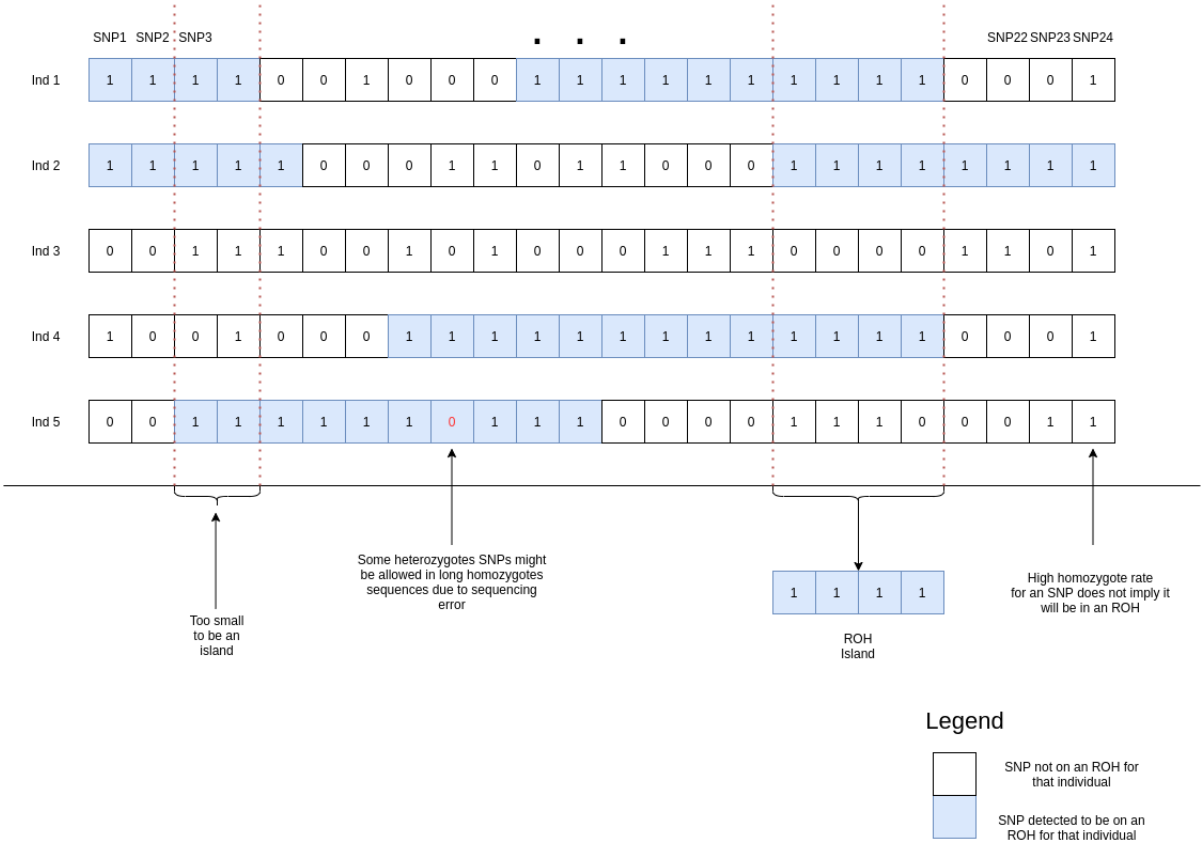


Figure 4.1: *Schematization for the detection of ROH islands. The blue blocks are segments identified as ROH for individuals. The block of 4 SNPs displayed below is a representation of what would be identified as a ROH island.*

the important aspects of the problem are:

(a) ROH is an individual characteristic. Some individuals will have more than one ROH; some will have ROH of great lengths; some might not have ROH at all. Their size and number may vary greatly. ROH islands, on the other hand, is related to the population;

(b) Some degree of heterozygosity, or in other words 0's, might be allowed inside ROH due to sequencing error;

(c) There is a "minimum size" to call a sequence a ROH and a region a ROH island. Breaking up the array in very small blocks of SNPs might not aid the geneticist in his analysis;

(d) To identify ROH, the probability of having an almost uninterrupted sequence of 1's passing through the SNP is more important than the probability of that individual SNP being homozygote. Hence, the proportion of homozygosity inside a ROH island might be lower than in other regions.

Taking these aspects into consideration, we can start posing the problem as a change point detection problem.

Data transformation Instead of modeling the probability of each SNP being homozygote, we can transform the data so that we model the probability of a sequence of homozygotes SNPs contain that SNP. The transformation is similar to a moving average. For each individual we do:

(i) Select an integer r that will be the radius and α a homozygote threshold;

(ii) Create a new SNP array for that individual;

(iii) For the i-th SNP between r and $m-r$, compute the mean $\hat{\mu}_i$ of the data on the interval $[i-r, i+r]$ in the original array. For $i < r$, take the interval $[i, i+2r]$ and for $i > m-r$, take $[i-2r, i]$;

(iv) If $\hat{\mu}_i > \alpha$, then assign 1 to the i-th SNP in the copy. Else, assign 0 .

The idea of the transformation is to take into account observations **(b)** and **(d)**. Of course, several other transformations are possible and might yield similar results.

Choosing a very high value of r turns the transformed data into a massive array of 0's, and hence we lose almost all our information. It is best to select small values so that we do not lose much information but still contribute towards the idea of modeling sequences of homozygosity.

Change point detection Using the transformed data, we can use binary change point detection method to segment the data into blocks that have different probability parameters. The parameter is interpreted as the probability of a homozygote sequence of SNPs passing through that specified SNP. The reason we want to do this segmentation is that we work with the following hypothesis: **ROH islands will be detected in homogenous blocks in the SNP array with a higher probability of containing homozygote sequences of SNPs than other regions**. Hence, the main idea will be to segment the data and search for the blocks with the highest probabilities.

Regularization Choosing a regularization that suits our purpose is not an easy task. We could impose the condition **(c)** by defining a loss function that assigns an infinite loss to any segmentation that has a block shorter than a prespecified size.

How to chose this size? One approach is to specify the number of SNPs allowed in a block, hence compute $|s-r+1|$ and check if the size is greater than the specified value. However, this can be problematic. SNPs have physical locations on the chromosome measured in bases, as referring to the number of base pairs since the start of the chromosome. ROH and ROH islands are usually measured in Kb, Kilobases, or Mb, Megabases.

Moreover, the distance between SNPs is non-uniform across the chromosome. The distance between SNP1 to SNP2 is different from SNP2 to SNP3. Hence, two blocks containing the same number of SNPs might differ greatly in actual physical size.

We can decide that a block is too short if $|L(s) - L(r)|$ is smaller than a specified value in Kb or Mb, where L is the SNP physical location. The specified value has a more natural interpretation and might be easier to choose.

We can also penalize blocks that are slightly above the threshold to create more homogeneous blocks. The penalization function for each block was

$$\rho(r,s) = \begin{cases} +\infty & \text{, if } \frac{|L(s)-L(r)|}{\beta} \leq T \\ \frac{\sqrt{n}}{|L(s)-L(r)|/\beta} & \text{, otherwise.} \end{cases}$$

T is the physical distance threshold we are considering, and $\beta = 10^6$ is a scaling factor to work on Mb unit.

4.1.4 Dataset

The data was obtained from Human Genome Diversity Project (HGDP) database, genotyped for a set of approximately 600,000 SNP markers from Illumina HuHap 650k platform Li et al. [2008]. We considered SNPs from chromosome 22 of African, Asian, and European populations. Each row represents an individual from that population, and the columns are a codification of that individual SNP array.

For each SNP position, we have what is called a reference allele. The entries of consist of three possible values: 0, 1, and 2, which code for the number of alleles that are identical to the reference allele. Since we are interested in homozygosity, we recodify the entries: 0 and 2 from the original dataset are mapped to 1, meaning that the individual is homozygote for that SNP, and 1 from the original data set is mapped to 0, representing that the individual is heterozygote for that SNP.

All the graphics are shown in this section is for the African population. The graphics for the other populations can be found in appendix D.

4.2 ROH island detection with PLINK

In the simulation, we could verify model performances because we had the ground truth. Since this is not available for the application, we compare the results of our model with the reference method of identifying ROH and ROH Islands.

We performed ROH identification in the same dataset using the software PLINK v1.9 Purcell et al. [2007], a well-established command line software designed to solve many medical and population genetics problems.

Given a SNP array, we can use PLINK be used to perform ROH identification across the genome of each individual and then retrieve the SNPs which appear the most in ROH. If SNPs are together in a window that is long enough, it is called a ROH island. First, we obtain the frequency at which SNPs appear in ROH, and then, by choosing a cutoff in the distribution of frequencies, we can determine which windows correspond to ROH islands. The cutoff is usually computed considering an upper quantile. **This quantile is computed considering the frequency distribution of SNPs in ROHs in all chromosomes but, for our example, we will use only chromosome 22.**

We ran PLINK using the same criteria described by McQuillan *et al.* McQuillan et al. [2008] and by Kirin *et al.* Kirin et al. [2010]. Table 4.1 shows a summary of the data set after preprocessing and PLINK results.

In order to detect ROH islands, we chose the cutoff as the 99% percentile for all three populations. Figure 4.2 shows the ROH detected for the African population for each individual. Red blocks show the ROH Islands, regions that the frequency was above the cutoff.

Figure 4.3 plots the frequency that SNPs are in ROH. Figure 4.2 can be interpreted as counting the number of "green blocks" that pass through that SNP.

We can see regions in the middle of the SNP array that have a higher frequency of ROH. However, only one of those regions would be detected as a ROH Island given our 99% cutoff.

Population	NSNPs	Size	Pop with ROH(%)	NROH	Avg size (Mb)	Max size (Mb)
AF	7943	176	65.90	196	1.227	13.25
EA	7977	218	86.69	416	0.866	8.30
EUR	7945	157	73.25	230	0.855	5.06

Table 4.1: *A summary of the ROH on chromosome 22 detected using PLINK. The third column displays the percentage of individuals in the population that have at least one ROH, and the fourth shows the number of ROHs detected in the population. The last two columns show the average ROH size and the maximum ROH size in Mega bases, respectively. We only considered chromosome 22. The number of SNPs differ in populations due to missing data cleaning.*

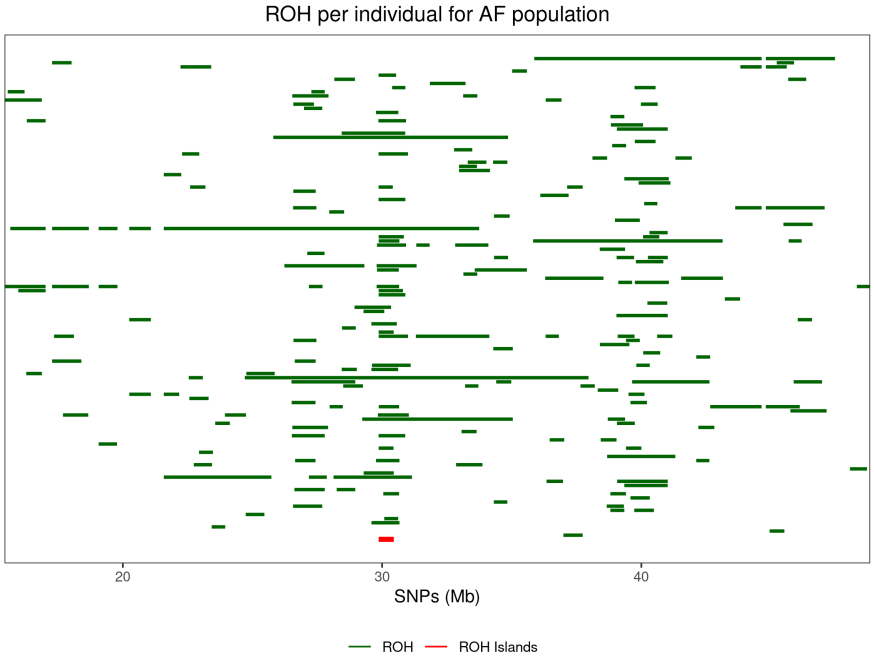


Figure 4.2: *SNPs plot for the African population. The green lines represent detected ROH for an individual. Lines at the same height represent blocks for the same individual. Not all individuals have detected ROH.*

4.3 Comparison

We end the Chapter comparing the two approaches. For our modelling, we fitted the hierarchical segmentation with $\lambda = 1$. The data transformation parameters were $r = 5$ and $\alpha = 0.95$. The regularization function parameters were $T = 1$ and $\beta = 10^6$.

Figure 4.4 shows the frequency plot of PLINK along with a block plot for hierseg. The right purple scale shows the probability parameter of the block.

It seems that regions detected with higher frequencies of ROH detected by PLINK were segmented as different blocks by our model and have higher probability parameters. For the African population, the block with the highest probability parameter corresponds to the detected ROH Island. However, this does not hold for the other populations. The blocks still have higher probability parameters than the average, but their ranks are not the same.

Table 4.2 shows the islands detected by PLINK for the three populations.

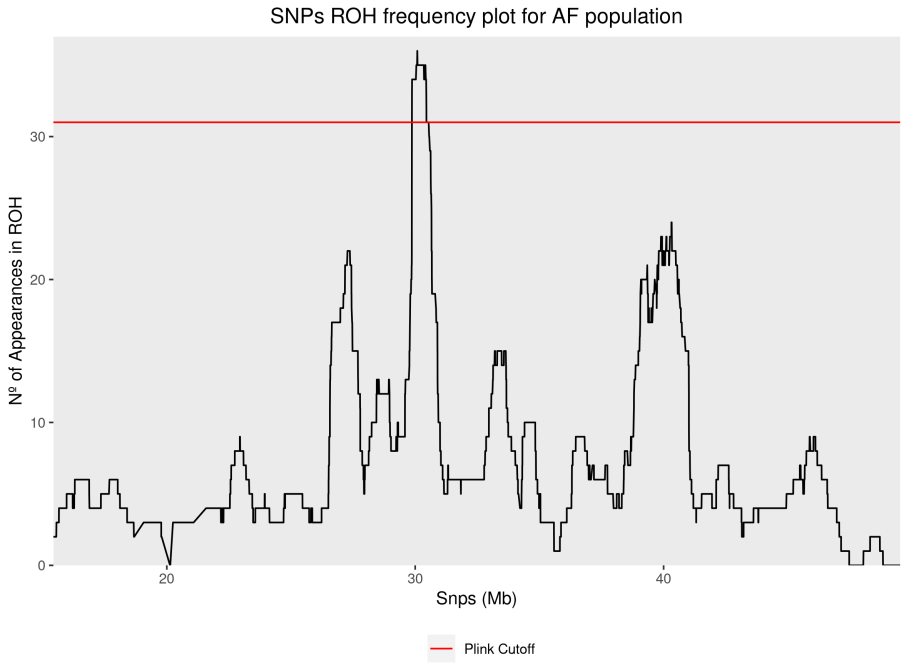


Figure 4.3: *Frequency plot of each SNP for the African population. There are some "spikes", which indicate that SNPs in that region have high frequency rate on ROHs for that population. The red line is the 99% percentile cutoff, so that SNPs with frequency higher than that would be considered as an ROH Island.*

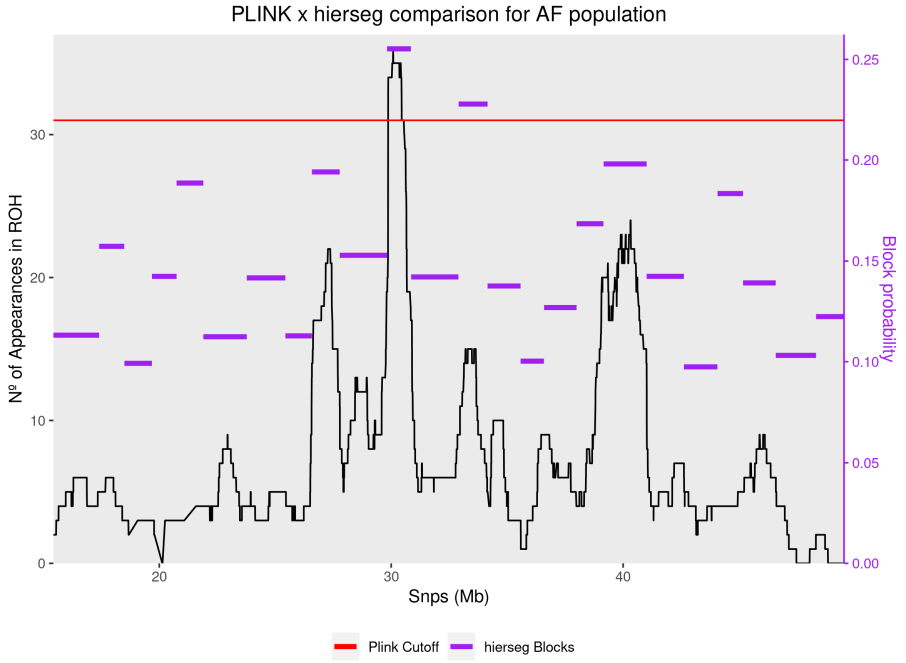


Figure 4.4: *Frequency plot of each SNP for the African population along with block plot for the hierseg model. The red cutoff line is for PLINK only.*

Population	Begin (Mb)	End (Mb)	NSNPs	Rank HS
AF	29.86919	30.45203	72	1
EA	39.62168	40.54563	77	6
EUR	30.00774	30.56019	62	3

Table 4.2: *Islands detected by PLINK for each of the populations, with the name of the beginning and ending SNP. The NSNPs column shows the number of SNPs in the Island. The rank HS column shows which position the block holds if we order decreasingly by the probability parameter.*

Chapter 5

Conclusions

In this work, we have presented the change point detection problem, defining it mathematically and viewing briefly some applications. We focused on the offline binary change point problem, with multiple and unknown change points, studying regularized likelihood methods and their consistency.

For the computation of the estimators, we showed that dynamical programming, hierarchical segmentation and disciplined convex programming can be applied. We provided pseudocodes for the estimators, and performed simulations studies to answer some raised questions. We provided an R package to fit all the models proposed in this work.

Finally, we tackled the problem of detecting ROH Islands in genetics, explaining the basic idea behind the problem and one possible way to frame it as a change point detection. We compared it to PLINK and obtained results that are related, but not identical.

Appendix A

Consistency Proofs

We will focus on proving that it the PML estimator is strongly consistent. In order to prove the theorem, we will establish some notation and prove two lemmas.

Suppose the data was generated by an ordered set $C^* = \{c_1^*, \dots, c_k^*\}$ and $\mathbf{p}^* \in [0,1]^{k+1}$.

Let $p \in [0,1]$, consider the function

$$q(p) = (1-p)\log(1-p) + p\log(p) \quad ,$$

defining $q(0) = 0 = q(1)$. The function $-q(p)$ is the entropy of binary distribution with parameter p.

It is easy to check that $q \in C^\infty$ and is strictly convex. Let I be an integer interval of $1 : (m-1)$, and define

$$Q(I; \mathbf{X}) = n|I| q(\hat{p}_I) \quad ,$$

where $\hat{p}_I = \frac{\sum_{i=1}^n \sum_{c \in I} x_c^{(i)}}{n|I|}$. For a set C with $\hat{\mathbf{p}}_C = (\hat{p}_1, \dots, \hat{p}_k, \hat{p}_{k+1})$, the log-likelihood can be written as

$$\begin{aligned}
\hat{l}(C; \mathbf{X}) &= \sum_{i=1}^{n} \sum_{j=0}^{k} \sum_{c=1+c_j}^{c_{j+1}} (1 - x_c^{(i)})\log(1-\hat{p}_j) + x_c^{(i)}\log(\hat{p}_j) \\
&= \sum_{j=0}^{k} n|c_{j+1} - c_j| q(\hat{p}_j) \\
&= \sum_{j=0}^{k} Q((1+c_j) : c_{1+j}; \mathbf{X}) \quad .
\end{aligned}$$

We proceed to state and prove the lemmas.

Lemma A.0.1. Let $I = r : s \subset 1 : (m-1)$ and suppose there exists $K \geq 1$ changepoints $\{c_1, \dots, c_K\}$ in $I \backslash \{s\}$. Then there exists a constant $\alpha > 0$ such that

$$\frac{\left[\sum_{k=1}^K Q(I_k; \mathbf{X})\right] - Q(I; \mathbf{X})}{n|I|} > \alpha \quad \text{a.s.} \quad ,$$

where $I_1 = r : c_1$, $I_k = (1 + c_k) : c_{k+1}$, $\forall\, k \in 2 : (K-1)$, and $I_K = (1 + c_K) : s$.

Proof. By the observations above we have that

$$\frac{\left[\sum_{k=1}^K Q(I_k; \mathbf{X})\right] - Q(I; \mathbf{X})}{n|I|} = \left[\sum_{k=1}^{K} \frac{|I_k|}{|I|} q(\hat{p}_{I_k})\right] - q(\hat{p}_I)$$

By the Law of Large Numbers, $\hat{p}_{I_k} \xrightarrow{\text{a.s.}} p_{I_k}$ for every k, and $\hat{p}_I \xrightarrow{\text{a.s.}} p_I$, where

$$p_I = \sum_{k=1}^{K} \frac{|I_k|}{|I|} p_{I_k} \quad .$$

Hence, by the Continuous Mapping Theorem, we have that

$$q(\hat{p}_{I_k}) \xrightarrow{\text{a.s.}} q(p_{I_k}),\ k \in 1:K, \quad \text{and} \quad q(\hat{p}_I) \xrightarrow{\text{a.s.}} q(p_I) \quad .$$

Since p_I is a convex combination of p_{I_k} and $p_{I_k} \neq p_{I_k+1}\, \forall\, k \in \{1, \ldots, K-1\}$, we can use the strictly convex property of q to obtain

$$\frac{\left[\sum_{k=1}^{K} Q(I_k; \mathbf{X})\right] - Q(I; \mathbf{X})}{n|I|} \xrightarrow{a.s.} \left[\sum_{k=1}^{K} \frac{|I_k|}{|I|} q(p_{I_k})\right] - q(p_I) = \alpha > 0 \quad .$$

□

Lemma A.0.2. Let $I = r : s \subset 1 : (m-1)$ be an interval such that there are no changepoints in $I \backslash \{s\}$. Then

$$\min_{c \in I \backslash \{s\}} Q(r : c; \mathbf{X}) + Q((c+1) : s; \mathbf{X}) - Q(I; \mathbf{X}) < 8|I|\log(\log(n)) \quad .$$

Proof. Let p_I be the parameter that generated the data in the block, and

$$Y_i = \sum_{j=r}^{s} \left[\mathbf{1}_{\{X_c^i = 1\}} - p_I\right] \quad ,$$

$$Z_n = \sum_{i=1}^{n} Y_i \quad .$$

Observe that $\{Y_i\}_{i=1}^n$ are i.i.d., $\mathbb{E}[Y_1] = 0$, $\text{Var}(Y_i) = |I| p_I (1 - p_I)$. By the Law of the Iterated Logarithm we have that, $\forall \epsilon > 0$,

$$Z_n < (1+\epsilon)|I| p_I (1-p_I) \sqrt{2n\log(\log(n))} \quad \text{a.s.} \quad ,$$

and taking $\epsilon = 1$, it follows that

$$\frac{Z_n}{n|I|\sqrt{p_I(1-p_I)}} = \frac{(\hat{p}_I - p_I)}{\sqrt{p_I(1-p_I)}} < 2\sqrt{\frac{2\log(\log(n))}{n}} \quad \text{a.s.} \quad .$$

Given $c \in I \backslash \{s\}$, define $I_1 = r : c$ and $I_2 = (c+1) : s$. The result also holds switching I for either I_1 or I_2 in the equation above, since $p_I = p_{I_1} = p_{I_2}$. Since $\hat{p}_I$ is the Maximum likelihood estimator for I, we have that

$$\begin{aligned} Q(I; \mathbf{X}) = n|I| q(\hat{p}_I) &= \sum_{i=1}^{n} \sum_{j=r}^{s} (1 - x_c^{(i)}) \log(1 - \hat{p}_I) + x_c^{(i)} \log(\hat{p}_I) \\ &> \sum_{i=1}^{n} \sum_{j=r}^{s} (1 - x_c^{(i)}) \log(1-p) + x_c^{(i)} \log(p) \\ &= n|I| \left[(1 - \hat{p}_I)\log(1-p) + \hat{p}_I \log(p)\right] \quad , \end{aligned}$$

hence

$$-Q(I; \mathbf{X}) < -n|I| \left[(1 - \hat{p}_I)\log(1-p) + \hat{p}_I \log(p)\right] \quad .$$

Note that

$$\begin{aligned} Q(I_1;\mathbf{X}) + Q(I_2;\mathbf{X}) - Q(I;\mathbf{X}) &= n[|I_1|q(\hat{p}_{I_1}) + |I_2|q(\hat{p}_{I_2}) - |I|q(\hat{p}_I)] \\ &< n[|I_1|q(\hat{p}_{I_1}) + |I_2|q(\hat{p}_{I_2}) - |I|((1-\hat{p}_I)log(1-p) + \hat{p}_I log(p))] \\ &= n|I_1|\left[(1-\hat{p}_{I_1})\log\left(\frac{1-\hat{p}_{I_1}}{1-p}\right) + \hat{p}_{I_1}\log\left(\frac{\hat{p}_{I_1}}{p}\right)\right] + \\ &+ n|I_2|\left[(1-\hat{p}_{I_2})\log\left(\frac{1-\hat{p}_{I_2}}{1-p}\right) + \hat{p}_{I_2}\log\left(\frac{\hat{p}_{I_2}}{p}\right)\right] \quad . \end{aligned}$$

Using the fact that $\log(x) \leq x - 1, \forall\, x \in \mathbb{R}$, we have

$$\begin{aligned} n|I_1|\left[(1-\hat{p}_{I_1})\log\left(\frac{1-\hat{p}_{I_1}}{1-p}\right) + \hat{p}_{I_1}\log\left(\frac{\hat{p}_{I_1}}{p}\right)\right] &\leq n|I_1|\left[(1-p_{I_1})\left(\frac{1-p_{I_1}}{1-p} - 1\right) + p_{I_1}\left(\frac{p_{I_1}}{p} - 1\right)\right] \\ &= n|I_1|\left[\frac{(p_{I_1}-p)^2}{p(1-p)}\right] \quad . \end{aligned}$$

The same argument can be made for I_2. Therefore

$$\begin{aligned} Q(I_1;\mathbf{X}) + Q(I_2;\mathbf{X}) - Q(I;\mathbf{X}) &\leq n|I_1|\left[\frac{(p_{I_2}-p)^2}{p(1-p)}\right] + n|I_2|\left[\frac{(p_{I_2}-p)^2}{p(1-p)}\right] \\ &< n|I_1|\frac{4\log(\log(n))}{n} + n|I_2|\frac{4\log(\log(n))}{n} \\ &= 8|I|\log(\log(n)) \quad \text{a.s.} \quad . \end{aligned}$$

Since the equation above holds for every $c \in r : (s-1)$, the result is established. □

We proceed to prove the Theorem for the PML estimator.

Proof (Theorem 2.2.2)**.** First we will prove that the PML estimator $\hat{C}$ will almost surely contain C^*. Consider $C \subset 1 : (m-1)$ such that $C^* \not\subseteq C$. Then there exists $c^*_{j_1} \dots c^*_{j_s}$ in C^* that do not belong to C and are between c_{i-1} and c_i for some i. Define $C' = C \cup \{c^*_{j_r}\}_{r=1}^s = \{c_1, \dots, c_{i-1}, c^*_{j_1} \dots, c^*_{j_s}, c_i, \dots, c_s\}$. Let $I'_1 = c_{i-1} : c^*_{j_1}$, $I'_r = (1 + c^*_{j_r}) : c^*_{j_{r+1}}$, $I'_s = (1 + c^*_{j_s}) : c_i$ and $I' = c_{i-1} : c_i$. Then, by lemma A.0.1, the log likelihood difference satisfies

$$\hat{l}(C';\mathbf{X}) - \hat{l}(C;\mathbf{X}) = \left[\sum_{r=1}^{s} Q(I'_r;\mathbf{X})\right] - Q(I';\mathbf{X}) \overset{a.s.}{>} \alpha|I'|n \quad .$$

Hence

$$PML(C';\mathbf{X}) - PML(C;\mathbf{X}) \overset{a.s.}{>} \alpha|I|n - \lambda(R(C') - R(C))J(n) \longrightarrow +\infty \quad .$$

The divergence to infinite follows from the fact that $J(n)$ grows at a slower rate than n. It follows that the PML estimator $\hat{C}$ will almost surely contain C^*.

We will now prove that it will converge exactly to C^*. Consider now C such that $C^* \subset C$ and that there exists $c_j \notin C^*$. Define $C'' = C \backslash \{c_j\}$, $I''_1 = c_{j-1} : c_j$ and $I''_2 = (c_j + 1) : c_{j+1}$. The log-likelihood difference is then

$$\begin{aligned} \hat{l}(C'';\mathbf{X}) - \hat{l}(C;\mathbf{X}) &= Q(I'';\mathbf{X}) - Q(I''_1;\mathbf{X}) - Q(I''_2;\mathbf{X}) \\ &\overset{a.s.}{>} -8|I|\log(\log(n)) \quad , \end{aligned}$$

where the inequality holds $a.s.$ because of **Lemma** A.0.2, since there are no change points in I''. Hence

$$PL(C''; \mathbf{X}) - PL(C; \mathbf{X}) \overset{a.s.}{>} -8|I|\log(\log(n)) - \lambda(R(C'') - R(C))J(n) \longrightarrow +\infty \quad ,$$

which follows from the fact that $(R(C'') - R(C)) < 0$ since the R is piecewise increasing and $\log(\log(n)) \in o(J(n))$. □

We present a basic lemma to help in the proof of the consistency of the HS estimator.

Lemma A.0.3. Let $f, g : \mathbb{R} \to \mathbb{R}$ be twice differenciable convex functions. If there exists a constant α such that

$$f(x) + g(x) = \alpha \,,$$

then f and g must be linear functions.

Proof. Differentiating both sides twice we obtain that

$$f''(x) + g''(x) = 0 \,.$$

Since both $f''(x) \geq 0$ and $g''(x) \geq 0$, then $f''(x) = 0 = g''(x)$ and the result follows. □

The proof of the consistency for the hierarchical segmentation algorithm is similar to the proof provided above.

Proof (Theorem 3.1.1)**.** The proof is divided in two parts; first, we show that, at any given possible scenario, the algorithm takes a correct choice almost surely; then, an induction argument will guarantee that the algorithm is consistent.

Part 1 We begin by recognizing the possible types of integer intervals that the algorithm can receive as input. For every integer interval $r : s$, the following trichotomy holds:

(i) There are no change points in $r : s$;

(ii) There is only one points in $r : s$;

(iii) There are multiple change points in $r : s$.

The correct decision for the algorithm in case (i) is to halt, not performing more recursive calls for that interval; in any of the other two cases, the correct decision is to choose any of the change points available and perform recursive calls on the sub intervals.

Next, we provide a proof that the algorithm will take the correct decision almost surely for cases (i) and (iii). The proof for case (ii) is very similar to case (iii) and is omitted.

(i) Suppose $r : s$ has no change points inside it and let $\hat{C}_{HS}$ be the HS estimator, $I_1 = r : u$, $I_2 = (1 + u) : s$. We have

$$\begin{aligned} \tilde{Q}(I; \mathbf{X}) - \tilde{Q}(I_1; \mathbf{X}) - \tilde{Q}(I_2; \mathbf{X}) &= -\left[Q(I; \mathbf{X}) - Q(I_1; \mathbf{X}) - Q(I_2; \mathbf{X})\right] + \\ &\quad + J(n)\lambda\left[\rho(r, s) - \rho(r, u) - \rho(u + 1, s)\right] \\ &\overset{a.s.}{<} 8\log(\log(n)) + J(n)\lambda\left[\rho(r, s) - \rho(r, u) - \rho(u + 1, s)\right] \quad . \end{aligned}$$

$$8\log(\log(n)) + J(n)\lambda\left[\rho(r, s) - \rho(r, u) - \rho(u + 1, s)\right] \longrightarrow -\infty \quad ,$$

because $[\rho(r, s) - \rho(r, u) - \rho(u + 1, s)] < 0$ since R is piecewise increasing and $\log(\log(n)) \in o(J(n))$. Hence, no splitting will be done almost surely, and the algorithm will not perform more recursive calls.

(iii) Now we have to prove that **the algorithm will almost surely split at change points only**. Let $c_1^*, \ldots, c_k^*$ be change points in $r:s$ with parameters $(p_1^*, \ldots, p_{k+1}^*$, and define $I_j^* = c_{j-1}^* : c_j^*$ for $j \in \{1, \ldots, k+1\}$, remembering the convention $c_0^* = 0$, $c_{k+1}^* = s$. For $u \in r:s$. Consider

$$h(u) = \tilde{Q}(r:u; \mathbf{X}) + \tilde{Q}((u+1):s; \mathbf{X}) \quad ,$$

the loss of splitting at u. By the Strong Law of Large Numbers we have

$$h^*(u) := \lim_{n\to\infty} \frac{h(u)}{n} = -(u-r+1)q(p_{r:u}) - (s-u)q(p_{(u+1):s}) \quad ,$$

is well defined, where $p_{r:u} = \sum_{j=1}^{k+1} \frac{|I_j^* \cap r:u|}{|I_j^*|} p_j^*$ and $p_{(u+1):s} = \sum_{j=1}^{k+1} \frac{|I_j^* \cap (u+1):s|}{|I_j^*|} p_j^*$. If we can show that there exists a change point c^* such that

$$h^*(u) - h^*(c^*) > 0 \quad , \forall\, u \neq c^*,$$

then, as $J(n) = o(n)$, we have that eventually almost surely as $n \to \infty$

$$h(u) - h(c^*) = n\left(\frac{h(u) - h(c^*)}{n}\right) \overset{a.s.}{\geq} n(h^*(u) - h^*(c^*)) > 0 \quad ,$$

and therefore the algorithm will either split at c^* or not split. If we then show that it must split at c^*, then the result follows.

Lets focus on showing $h^*(u) - h^*(c^*) > 0$. First, extend the definition of $h^*(u)$ to all real numbers of on the interval $r:s$. On the integers, it has the same value as defined previously.

Given two consecutive change points c_{j-1}^*, c_j^*, we show that the $h^*(u)$ is concave on $[c_{j-1}^*, c_j^*]$. Let $u \in [c_{j-1}^*, c_j^*]$ and define $t(u) = \frac{c_{j-1}^* - r + 1}{u - r + 1}$. Then we can write

$$p_{r:u} = t(u) p_{r:c_{j-1}^*} + (1 - t(u)) p_j^* \,.$$

It is sufficient to show that $(u-r+1)q(p_{r:u})$ and $(s-u)q(p_{(u+1):s})$ are convex on this interval. Take $g(u) = (u-r+1)q(p_{r:u})$, the first derivative is

$$\begin{aligned} g'(u) &= q(p_{r:u}) + (u-r+1)t'(u)(p_{r:c_{j-1}^*} - p_j^*)q'(p_{r:u}) \\ &= q(p_{r:u}) - t(u)(p_{r:c_{j-1}^*} - p_j^*)q'(p_{r:u}) \,, \end{aligned}$$

where the second equality follows from the fact that $(u-s+1)t'(u) = -t(u)$. The second derivative of g is then

$$\begin{aligned} g''(u) &= \overbrace{t'(u)(p_{r:c_{j-1}^*} - p_j^*)q'(p_{r:u}) - t'(u)(p_{r:c_{j-1}^*} - p_j^*)q'(p_{r:u})}^{=0} \\ &\quad + (-t'(u)t(u))(p_{r:c_{j-1}^*} - p_j^*)^2 q''(p_{r:u}) \\ &\geq 0 \,, \end{aligned}$$

because $q''(p_{r:u}) > 0$ and $-t'(u)t(u) \geq 0$. We conclude that g is convex on $[c_{j-1}^*, c_j^*]$. An analogous argument shows that $(s-u)q(p_{(u+1):s})$ is also convex. This proves that h^* must be concave between change points. But remember that a concave function must attain a minimum at the end points an interval. Since $r:s$ can be partitioned by intervals of change points, h^* attains a global minimum at a change point. However, we still have to prove that any point u that is not change point does not attain the global minimum, so that the strict inequality holds. Since u must be in $[c_{j-1}^*, c_j^*]$ for some change points, it only attains the global minimum, if, and only if, h^* is constant on the interval.

Assume, by absurd, that h^* is constant in $[c^*_{j-1}, c^*_j]$. Then, for some α we must have that

$$-(u-r+1)q(p_{r:u}) - (s-u)q(p_{(u+1):s}) \;=\; \alpha \tag{A.1}$$

for all $u \in [c^*_{j-1}, c^*_j]$. Lemma A.0.3 implies that $(u-r+1)q(p_{r:u})$ and $(s-u)q(p_{(u+1):s})$ must be linear functions on $[c^*_{j-1}, c^*_j]$, therefore $q(p_{r:u})$ and $q(p_{(u+1):s})$ are constants. Since q is strictly convex and $p_{r:u}$ (respectively $p_{(u+1):s}$) is a convex combination of $p_{r:c^*_{j-1}}$ and p^*_j (respectively of $p_{c^*_j:s}$ and p^*_j), we conclude that $p_{r:u} = p^*_j = p_{(u+1):s}$ for all $u \in [c^*_{j-1}, c^*_j]$. Moreover we have that

$$p_{r:s} \;=\; \frac{1}{|I|}\big[(u-r+1)p_{r:u} + (s-u)p_{(u+1):s}\big] \;=\; p^*_j$$

and therefore the loss at any point is equivalent to the loss of not splitting. Repeating the argument for all intervals, we conclude that $p^*_i = p^*_j$ for all $i, j \in \{1, \ldots, k^*\}$, which is a contradiction since at least one change point exists. Therefore, the global minimum is achieved at a change point c^* and is not achieved at non change points. Moreover, this final argument shows that the loss at splitting is strictly greater than the loss at c^*. Therefore, it will almost surely split at a change point.

Part 2 We finish the proof using mathematical induction on the number of variables m.

If $m = 1$, then there is no possibility for change points, and the algorithm will not even have comparisons to make. Hence, the change point set is always empty by construction, and therefore is consistent.

Suppose that the algorithm is consistent for every $\tilde{m} <= m-1$, we will prove that it will be consistent for m.

The first run of the algorithm is on the interval $1:m$. By **Part 1**, if there are no change points, the algorithm will almost surely not split the interval and will halt since no other recursive calls will be made, and hence will be consistent.

If there are change points, take $c_0 = 0$ and $c_{|C^*|+1} = m$. **Part 1** tells us again that the algorithm a.s. takes the correct choice and splits at a change point c. After the split, recursive calls are made on $1:c$ and on $(c+1):m$. But the length of these arrays are at most $m-1$, and by induction hypothesis we have that the algorithm will a.s. retrieve all the change points in $1:c$ and in $(c+1):m$, obtaining then the whole change point set.

□

Appendix B

Rand Index proof

We prove the rand index formula provided in Chapter 3.2.

Proposition B.0.1. Let $C = \{c_1, c_2, \dots c_r\}$ and $C^* = \{c_1^*, c_2^*, \dots c_s^*\}$ be change point sets whose partitions are those described as above. Define $c_0 = c_0^* = 0$ and $c_{r+1} = c_{s+1}^* = m$. Then, the Rand Index is given by

$$R = 1 - \frac{\sum_{i=0}^{r}\sum_{j=0}^{s} M_{ij}|c_i - c_j^*|}{\binom{m}{2}} \quad ,$$

where

$$M_{ij} = \max\left(0, \min\left(c_{i+1}, c_{j+1}^*\right) - \max\left(c_i, c_j^*\right)\right) \quad .$$

Proof. Our strategy is to calculate the expression $a+b$ in the original formula by doing a partition of the pairs according to each element. For each x in $1 : (m-1)$ define A_x, the set of pairs (x, y) where $x < y$ and which the partitions agree, that is, they are in a single set in both partitions or in different sets in both partitions. Since this pair is being counted in the sum $a + b$ and each pair of $a + b$ is in some A_x, we have that

$$a + b = \sum_{x=1}^{m-1} |A_x| \quad .$$

The restriction $x < y$ avoids a pair of being counted twice.

First, we know that there are a total of $m - x$ pairs (x, y) of the form $x < y$. Let $c_i(x)$ and $c_j^*(x)$ be the smaller change points in C and C^* that are greater or equal than x so that $x \in (c_{i-1}(x) + 1) : c_i(x)$ and $x \in (c_{j-1}^*(x) + 1) : c_j^*(x)$. Then the partitions would agree on all pairs before $min(c_i(x), c_j^*(x))$, because they would put the elements of the pair in the same set, and they would agree after $max(c_i(x), c_j^*(x))$ because both of them would put the elements of the pair in different sets.

From this reasoning, there are a total of $|c_i(x) - c_j^*(x)|$ disagreements, hence

$$|A_x| = (m - x) - |c_i(x) - c_j^*(x)| \quad .$$

We can now substitute in the original equation

$$R = \frac{\sum_{x=1}^{m-1} |A_x|}{\binom{m}{2}} = 1 - \frac{\sum_{x=1}^{m-1} |c_i(x) - c_j^*(x)|}{\binom{m}{2}} \quad .$$

This is not yet what we wanted since $c_i(x)$ and $c_j^*(x)$ are functions of x, so we still have to check each element and find the change points associated to it. However, notice that if $x + 1 \leq c_i(x)$ and $x + 1 \leq c_j^*(x)$, then $c_i(x + 1) = c_i(x)$ and $c_j^*(x + 1) = c_j^*(x)$, and hence the term $|c_i(x) - c_j^*(x)|$

would appear again. Hence, for each pair $(c_i, c_j^*) \in C \times C^*$ (now they do not depend on x), we can associate a set $X_{ij} = \{x \in 1 : m\,;\, x \leq c_i,\, x \leq c_j^*\}$ and count how many elements this set has. If we define $M_{ij} = |X_{ij}|$, then we have that

$$\sum_{x=1}^{m-1} |c_i(x) - c_j^*(x)| = \sum_{i=0}^{r} \sum_{j=0}^{s} M_{ij} |c_i - c_j^*| \quad .$$

The number of elements x in X_{ij} is precisely the length of the interval

$$\max\left(c_i, c_j^*\right) : \min\left(c_{i+1}, c_{j+1}^*\right)$$

.

If this interval is ill defined, i.e. $\min\left(c_{i+1}, c_{j+1}^*\right) < \max\left(c_i, c_j^*\right)$, this just means that it is impossible for c_i and c_j^* to be both the greatest change points in an interval for any x, so their difference never appear in the summation. Hence, the number of times $|c_i - c_j^*|$ appear is

$$M_{ij} = \max\left(0, \min\left(c_{i+1}, c_{j+1}^*\right) - \max\left(c_i, c_j^*\right)\right) \quad .$$

□

Appendix C

R package bincpd

In order to make the methods studied here free to the general public, we created an R package named **bincpd**, short for binary change point detection, and made it available in github. This short section shows how to install, sample data from the distribution studied in this work and fit the models proposed using the package.

Installation

Installation is straightforward using the devtools package. Run the following command on **R** terminal.

```
devtools::install_github("https://github.com/Lucas-Prates/bincpd")
```

If you do not have devtools, install it by running the command below on the terminal.

```
install.packages("devtools")
```

Example

Simulation setup

Here we provide an example of usage of the package on simulated data. The function **rBBM** generates data according to the model proposed. Consider 50 data samples from a vector of size 20 separated in 4 blocks: $1:5$, $6:10$, $11:15$ and $16:20$. We assign the probability vector $(0.3, 0.7, 0.9, 0.1)$ to represent the probability of each of the blocks. Here is the code for sampling from the this setup.

```
library(bincpd)
set.seed(42)
sim_info <- rBBM(n = 50, m = 20, prob_vec = c(0.3, 0.7, 0.9, 0.1),
                 changepoints = c(5, 10, 15))
```

Notice that we have 4 blocks, but only 3 change points, which are precisely the end point of each block, with the exception of the last block. The functions returns us the **data_matrix**, the **changepoints** and the probability vector **prob_vec**. The last two are useful when we do not specify these arguments, and hence the change points and probabilities are simulated uniformly before simulating the data vector.

Estimators

A single function called **fit_model** wraps up all methods. The function takes 3 arguments: the **data_matrix**, the data that change point analysis will be done; the **method**, a string specifying the method to be fitted; and **arguments**, a list of arguments that are specific to that method. Check the documentation to see which arguments can be passed to which methods.

Five methods are provided: **hierseg**, **dynseg**, **cvdynseg**, **cvseg** and **fusedlasso**. All methods were implemented using Rcpp to speed up calculations. Below we show the code to fit each of the models using their default arguments.

```
sim_data <- sim_info$data_matrix
# Using default arguments for all methods.
hs_model <- fit_model(data_matrix = sim_data, method = "hierseg")
ds_model <- fit_model(data_matrix = sim_data, method = "dynseg")
cvds_model <- fit_model(data_matrix = sim_data, method = "cvdynseg")
cvseg_model <- fit_model(data_matrix = sim_data, method = "cvseg")
fused_model <- fit_model(data_matrix = sim_data, method = "fusedlasso")
```

Every model returns an S3 object of the class **bincpd**. Different models might return more or less information about their fit procedure/arguments, but all of them always contain at least four attributes:

changepoints : a list containing the set of estimated change points;

probabilities : a list containing the estimated probability parameters for each block;

loss : the final loss evaluated on the entire data set for the returned model;

n_cp : number of change points estimated.

Below, we show the change point set and probabilities estimated by each of the methods.

Model	Change points	Probabilities	Loss
hierseg	5 10 15	0.328 0.736 0.9 0.116	489.1267
dynseg	5 10 15	0.328 0.736 0.9 0.116	489.1267
cvdynseg	5 10 15 19	0.328 0.736 0.9 0.14 0.02	488.0979
cvseg	4 5 10 15 19	0.355 0.22 0.736 0.9 0.14 0.02	467.908
fusedlasso	1:19	*	472.1623

Table C.1: *Simulation output for the models. The fused lasso segmented every point as a change point, and we omitted the probability vector output.*

Albeit all models report the loss attained by the estimated model, they might not be comparable between different methods. For instance, the **hierseg** method and the **fusedlasso** have different loss functions, so loss comparison might not make sense.

Notice that the **fusedlasso** has over segmented the array. The reason is that the penalization constant default $\lambda = 1$ was too low. Here we exemplify how to pass arguments to a method, changing the λ arguments and also the **precision** argument. The precision is the threshold used by the **fusedlasso** to detect change points. When two consecutive probabilities differ by more than the precision, a change point is added to that location.

```
arguments <- list(lambda = 10, precision = 1e-2)
fused_model <- fit_model(data_matrix = sim_data, method = "fusedlasso",
                         arguments = arguments)
```

From this modification, the **fusedlasso** estimated the change points $5, 8, 10, 15, 19$, the probability vector was $0.364236, 0.7199909, 0.76, 0.8629977, 0.140014, 0.07283751$ and the loss was 472.1623.

Appendix D

Application Figures for other populations

Here, we provide the application graphics for the European and Asian populations. The same parameters where used both for PLINK and the hierarchical segmentation.

D.1 Europe

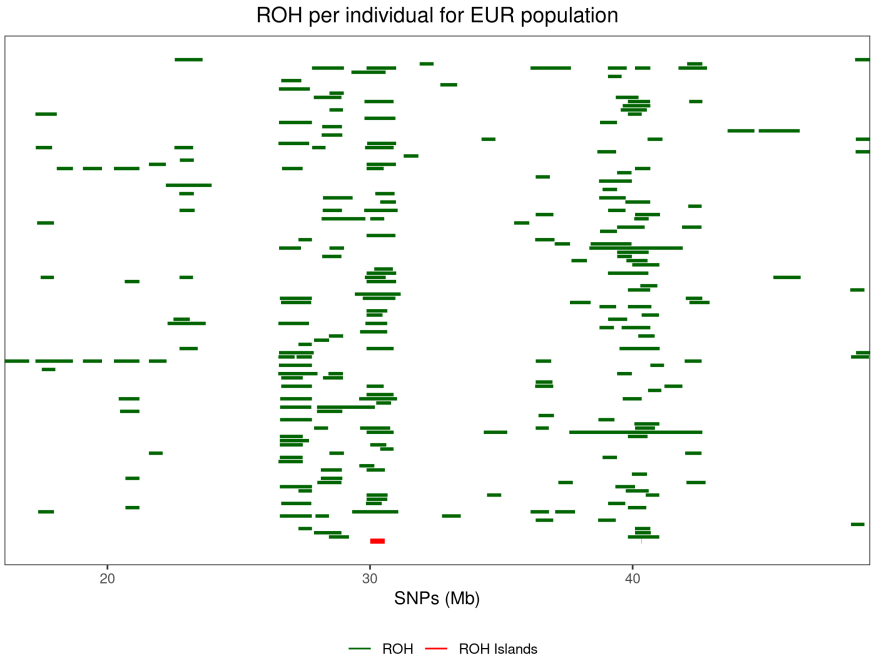


Figure D.1: *SNPs frequency distribution for the European population. The green lines represent detected ROH for an individual. Lines at the same height represent blocks for the same individual. Not all individuals have detected ROH.*

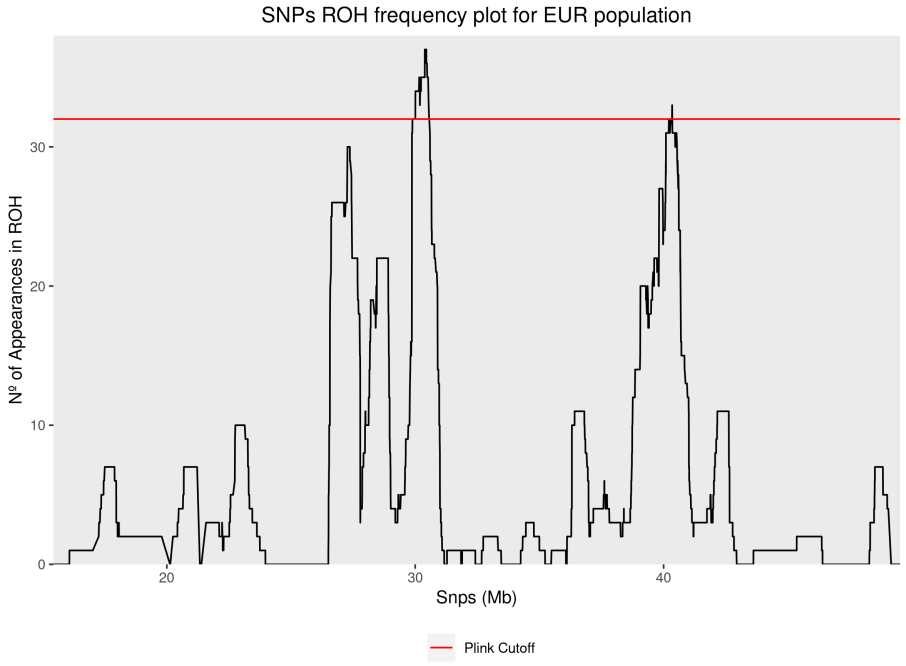


Figure D.2: *Frequency plot of each SNP for the European population. There are some "spikes", which indicate that SNPs in that region have high frequency rate on ROHs for that population. The red line is the 99% percentile cutoff, so that SNPs with frequency higher than that would be considered as an ROH Island.*

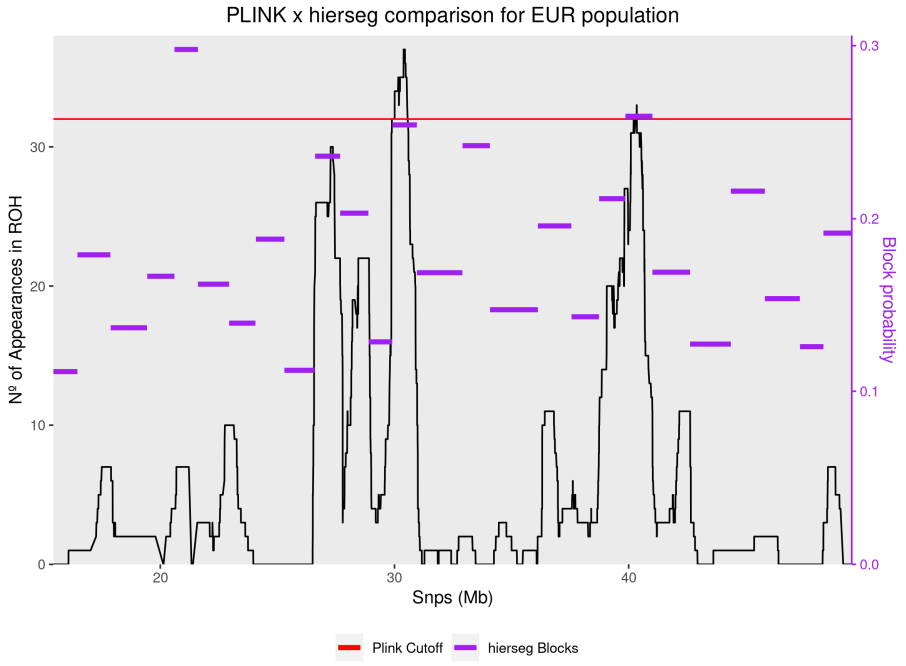


Figure D.3: *Frequency plot of each SNP for the European population along with block plot for the hierseg model. The red cutoff line is for PLINK only.*

D.2 Asia

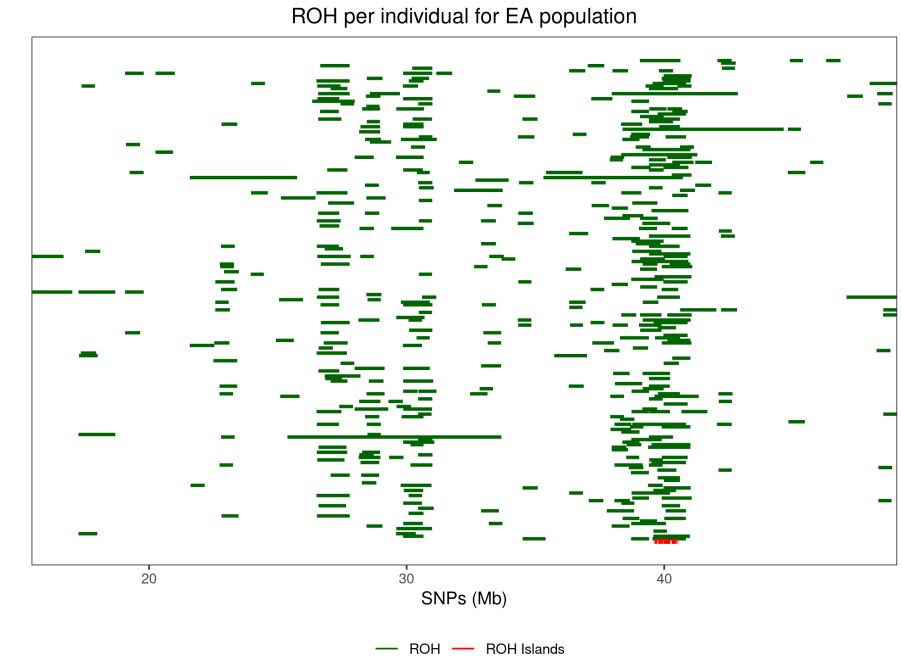


Figure D.4: *SNPs frequency distribution for the Asian population. The green lines represent detected ROH for an individual. Lines at the same height represent blocks for the same individual. Not all individuals have detected ROH.*

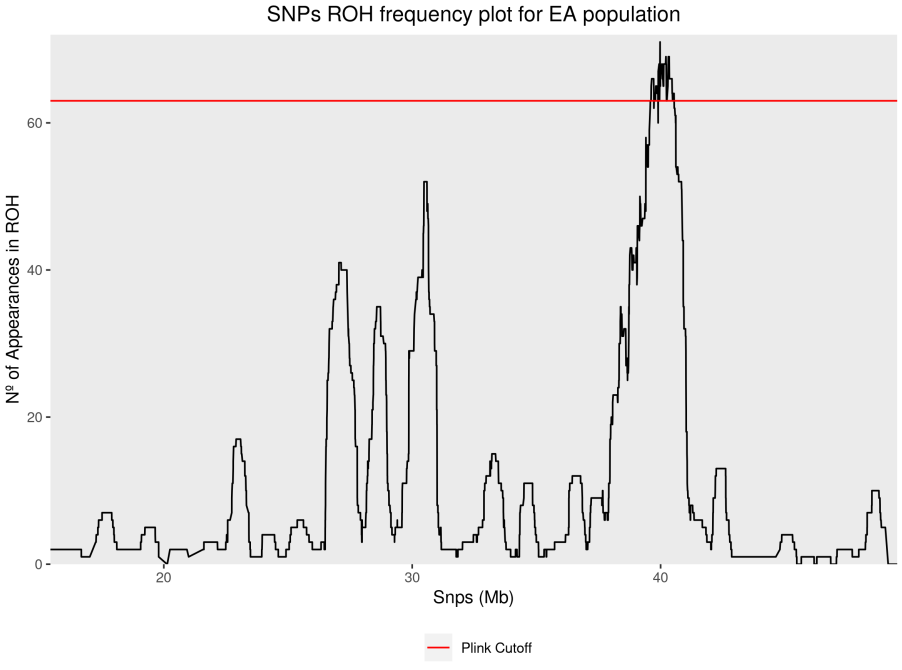


Figure D.5: *Frequency plot of each SNP for the Asian population. There are some "spikes", which indicate that SNPs in that region have high frequency rate on ROHs for that population. The red line is the 99% percentile cutoff, so that SNPs with frequency higher than that would be considered as an ROH Island.*

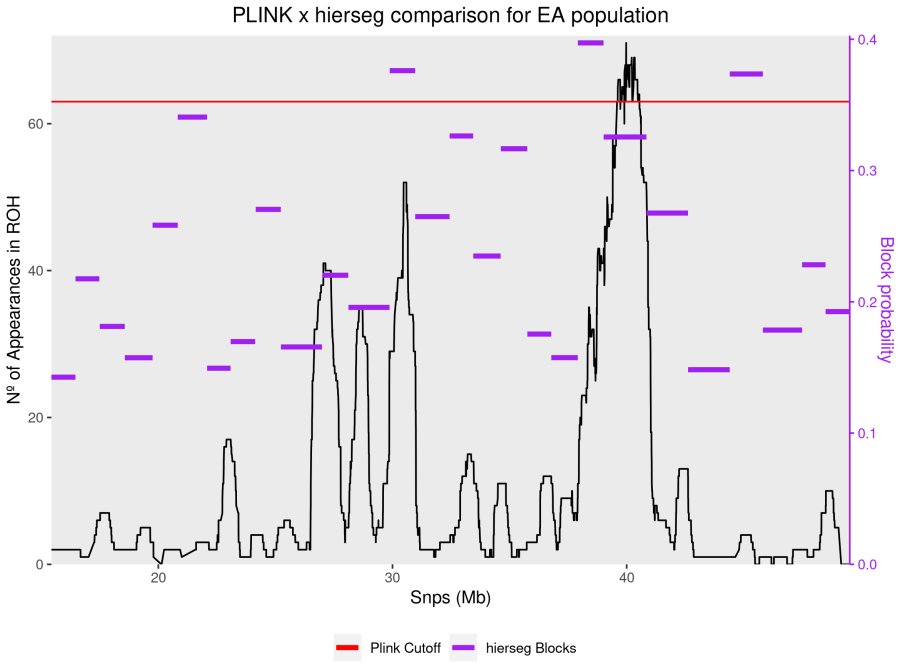


Figure D.6: *Frequency plot of each SNP for the Asian population along with block plot for the hierseg model. The red cutoff line is for PLINK only.*

Appendix E

Simulation Figures

k-10 m-100

CVSEG

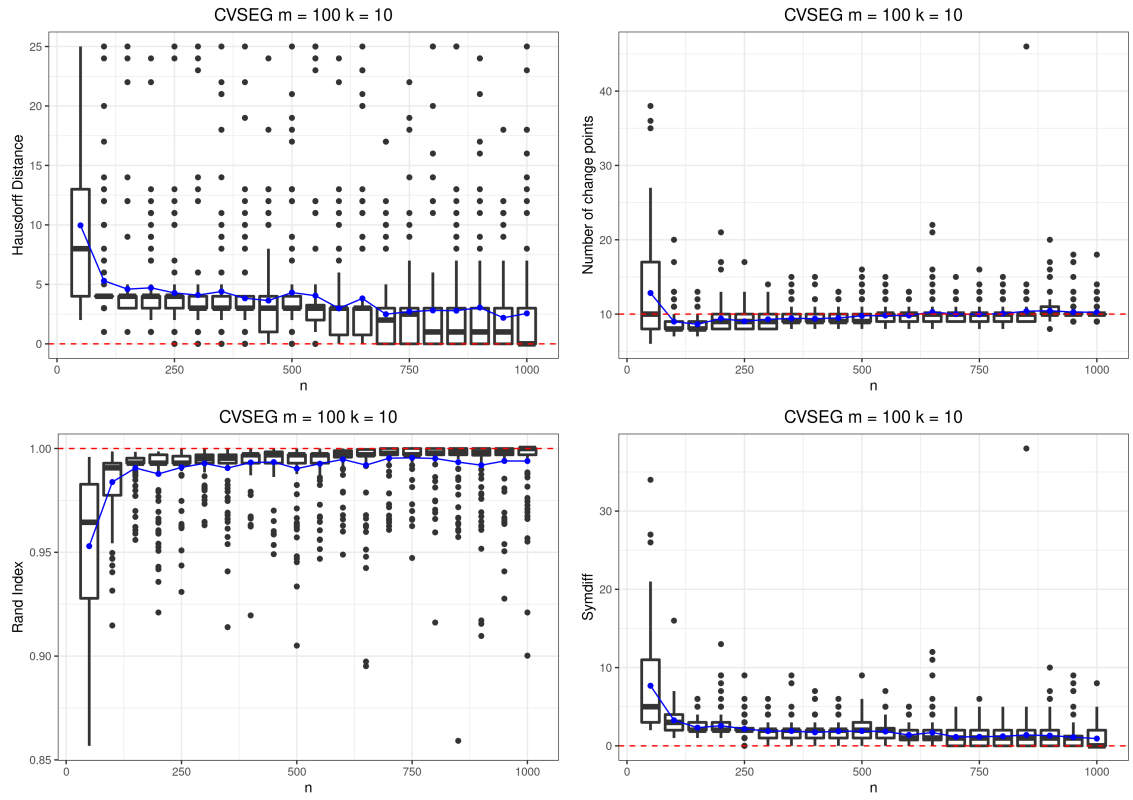


Figure E.1: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVSEG*

CVDPS

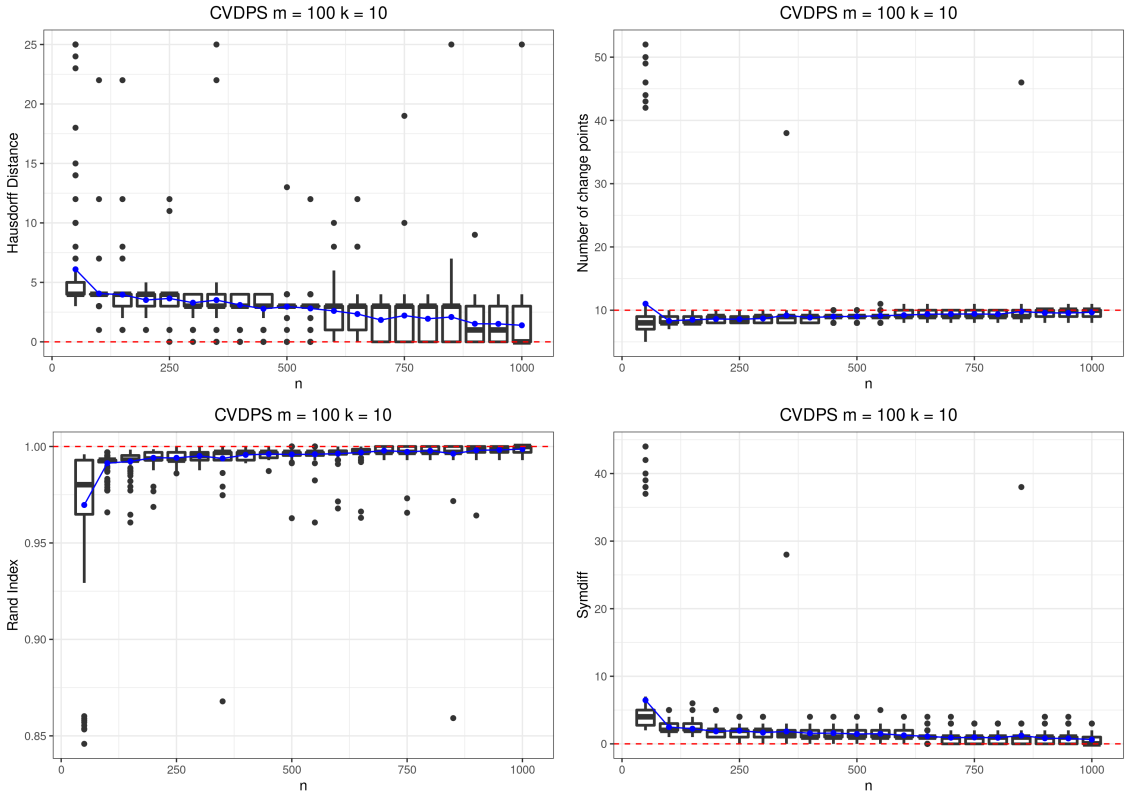


Figure E.2: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVDPS*

DPS LOG

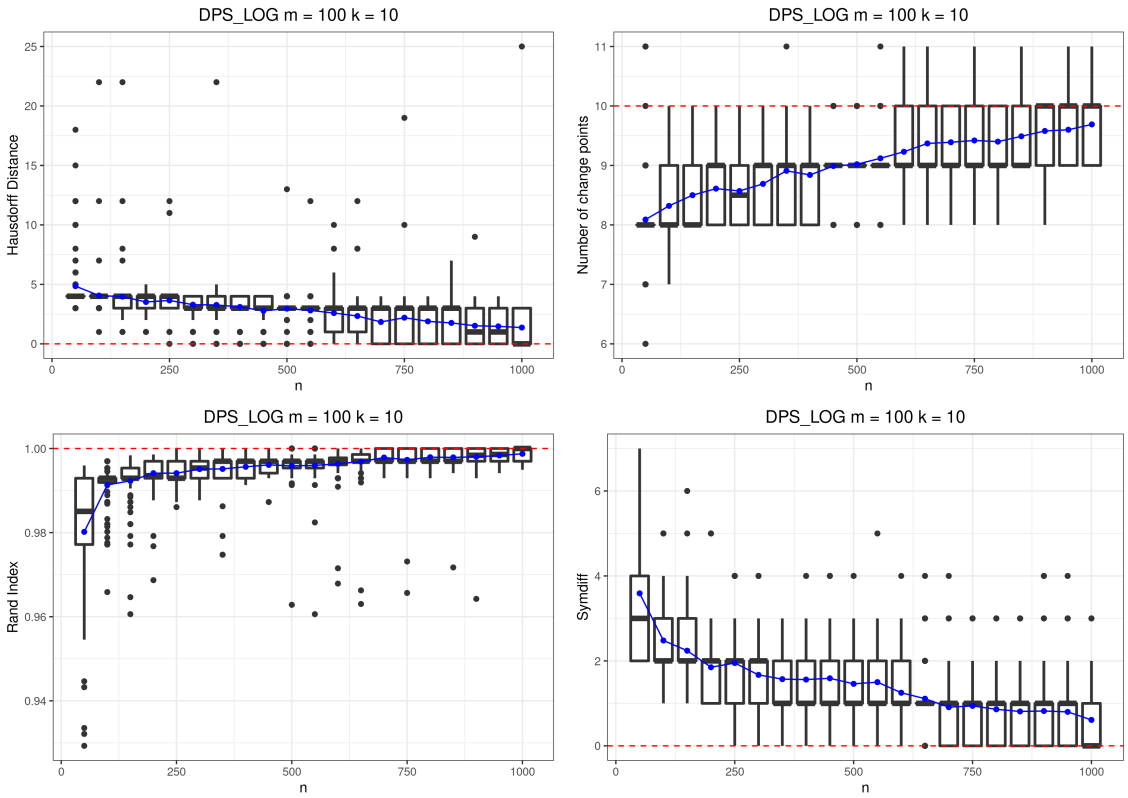


Figure E.3: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS LOG*

DPS SQRT

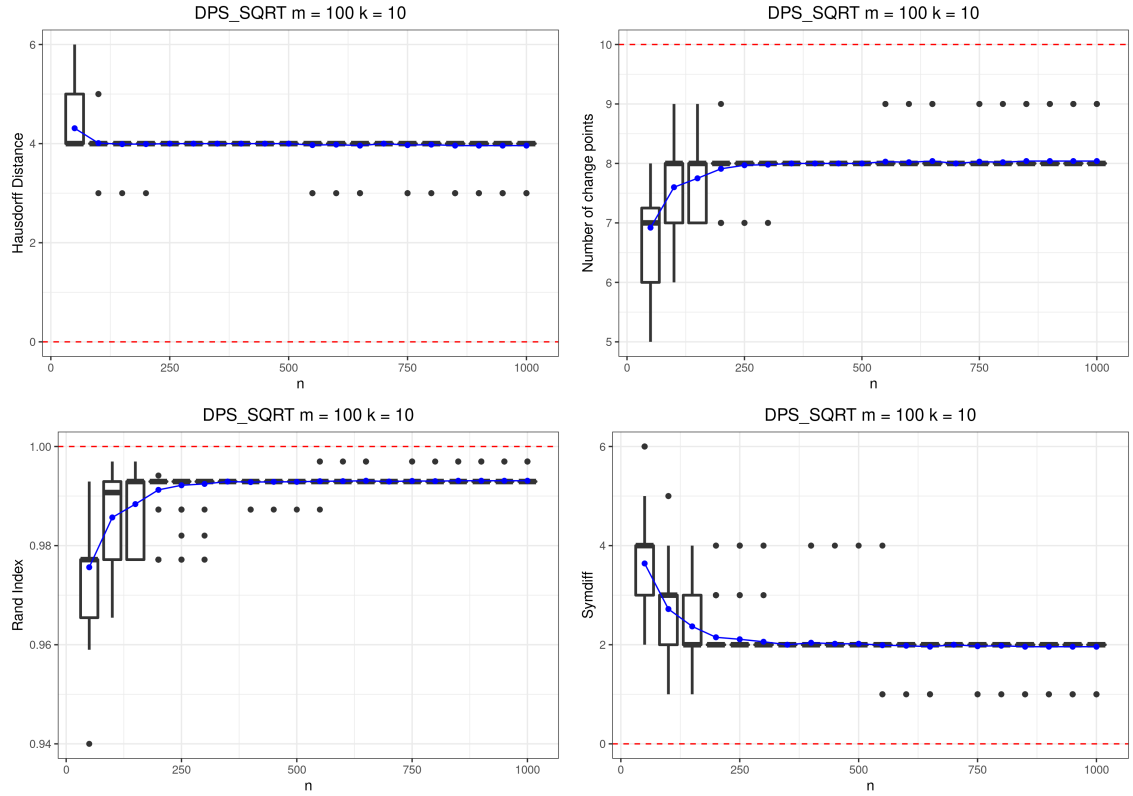


Figure E.4: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS SQRT*

HS

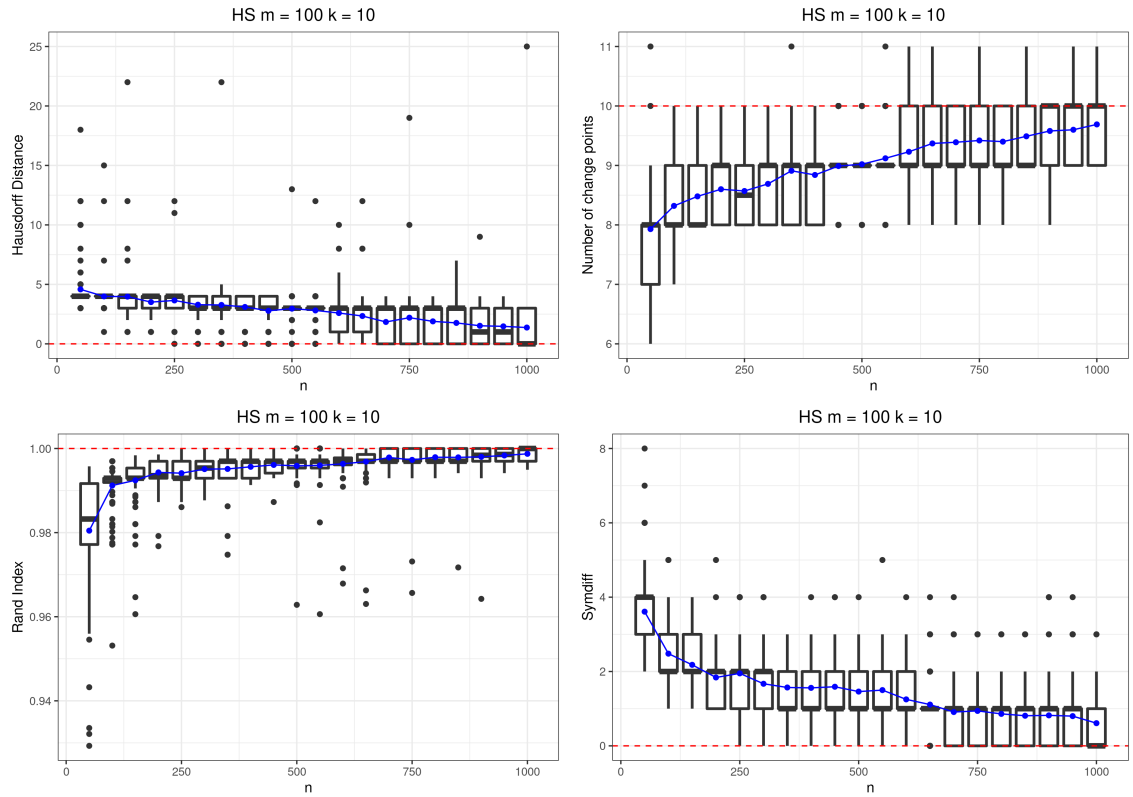


Figure E.5: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the HS*

k-10 m-200

CVSEG

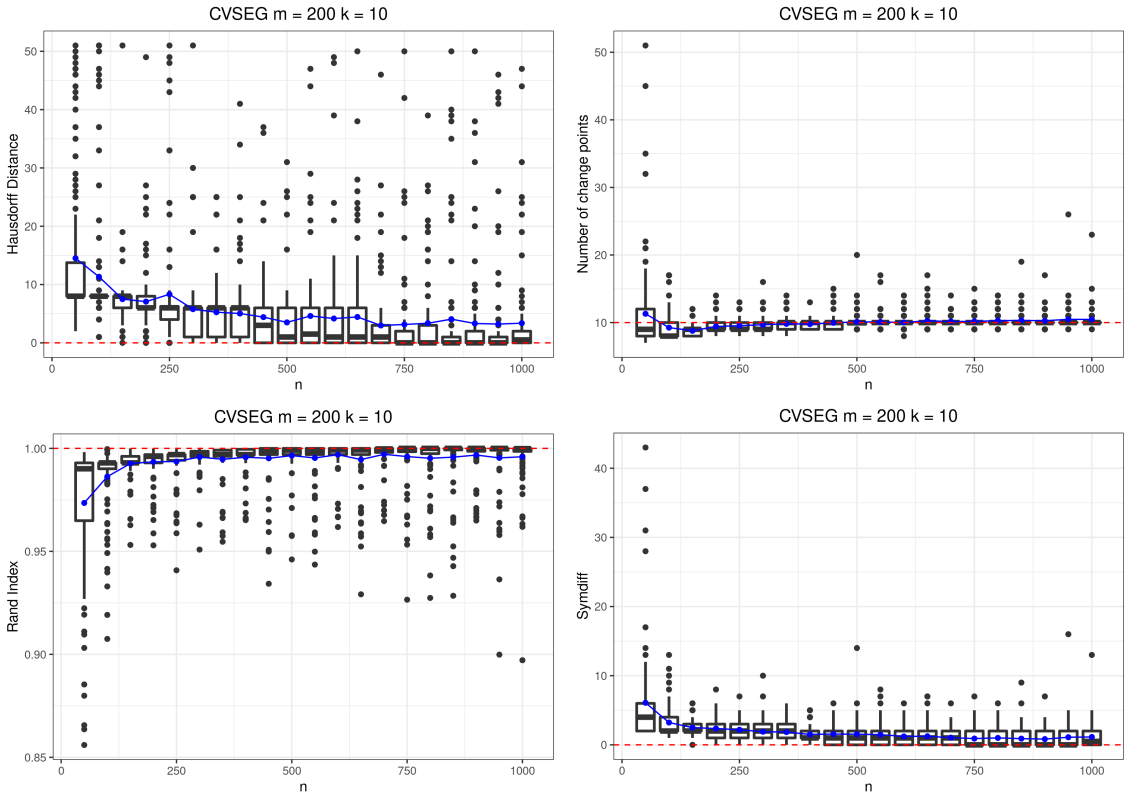


Figure E.6: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVSEG*

CVDPS

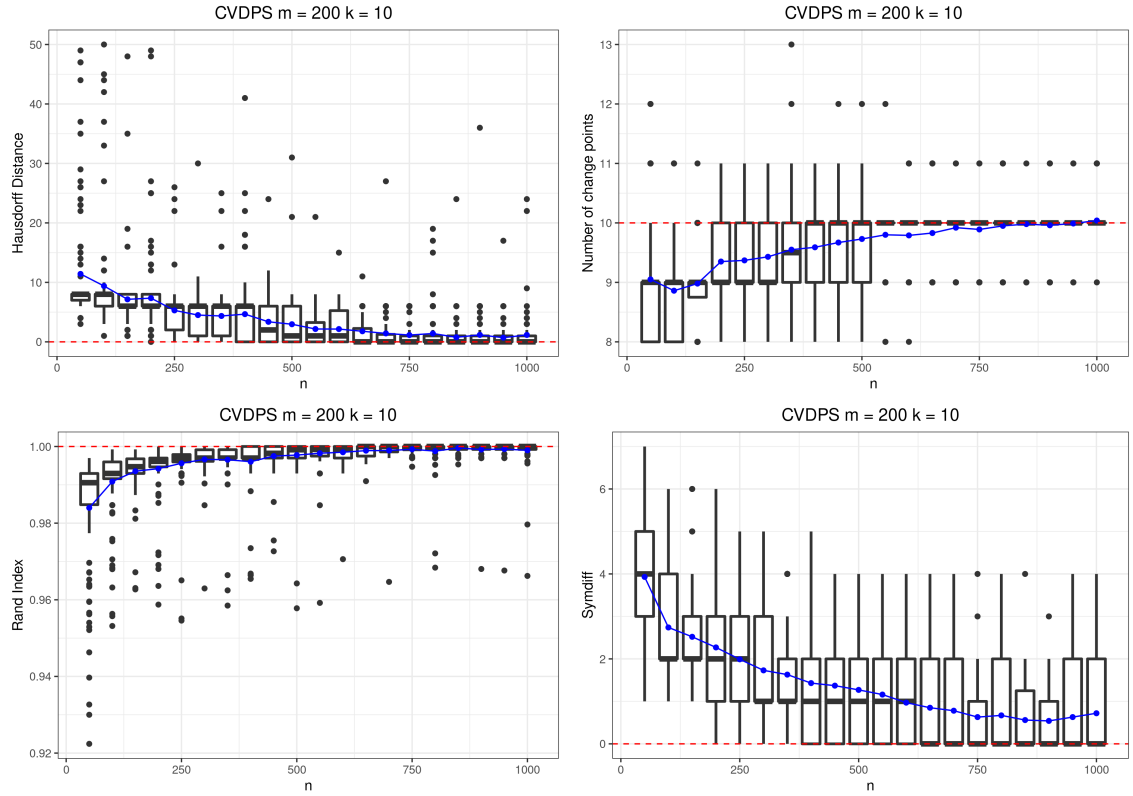


Figure E.7: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVDPS*

DPS LOG

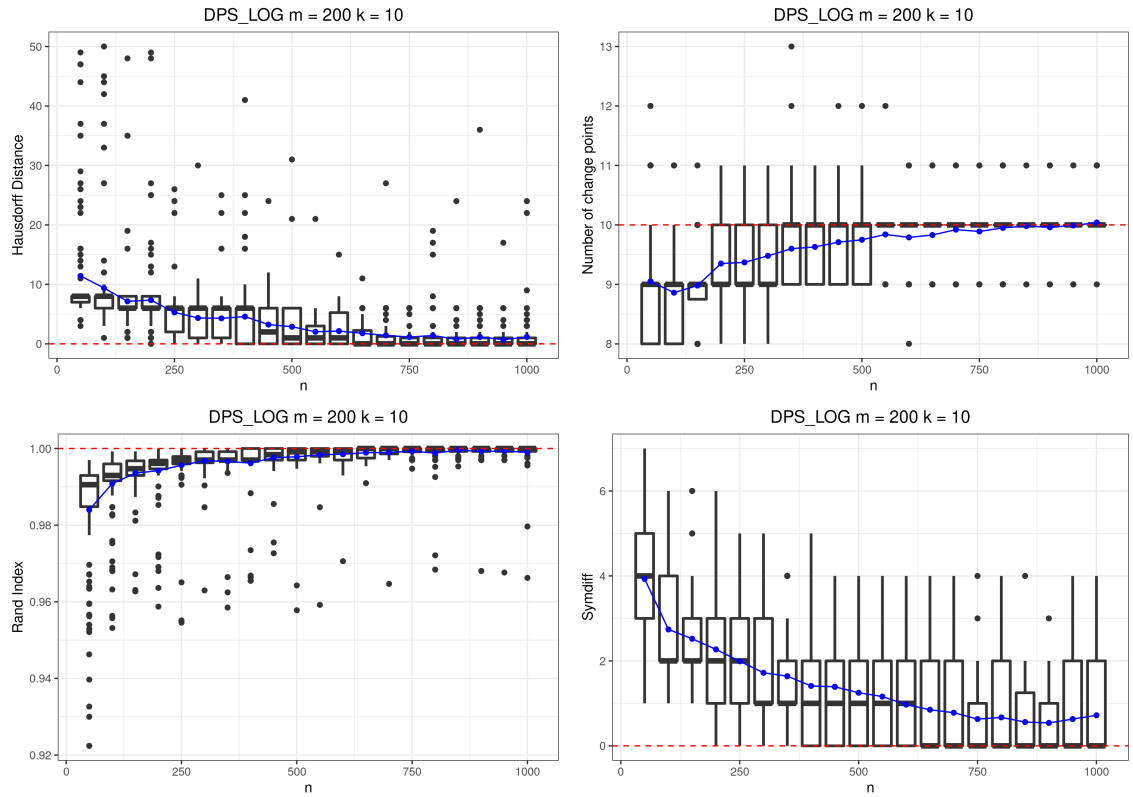


Figure E.8: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS LOG*

DPS SQRT

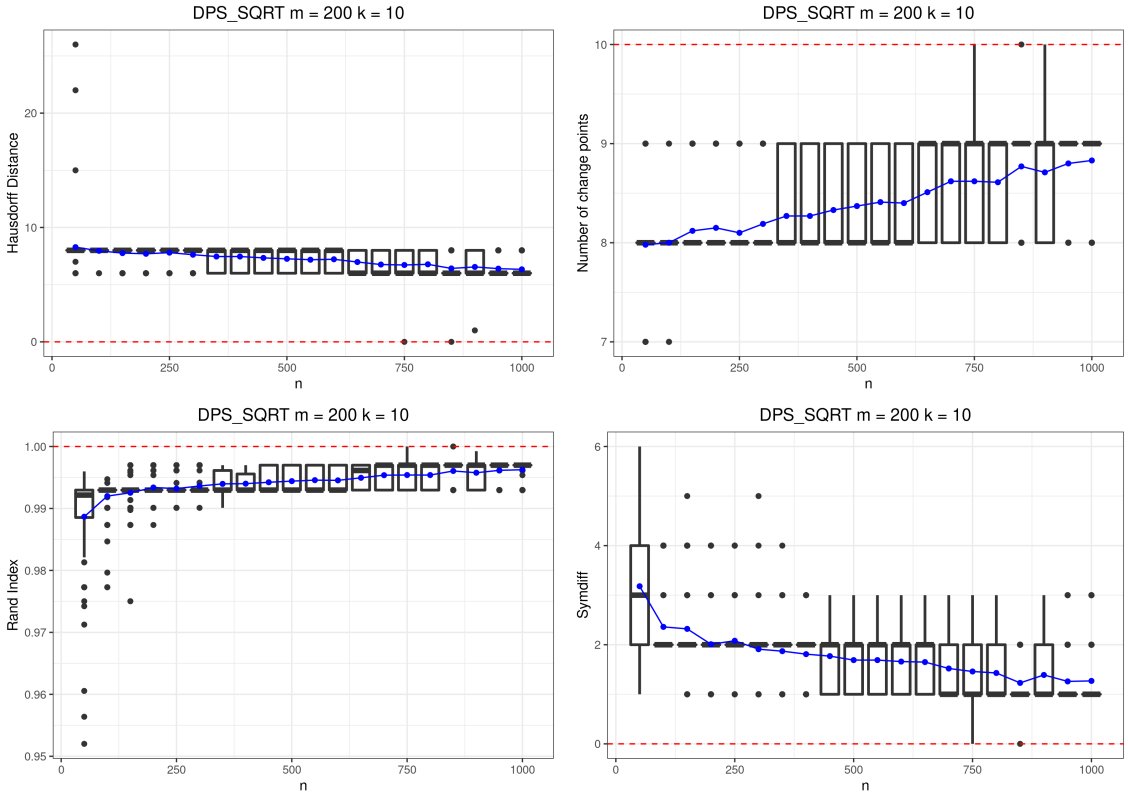


Figure E.9: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS SQRT*

HS

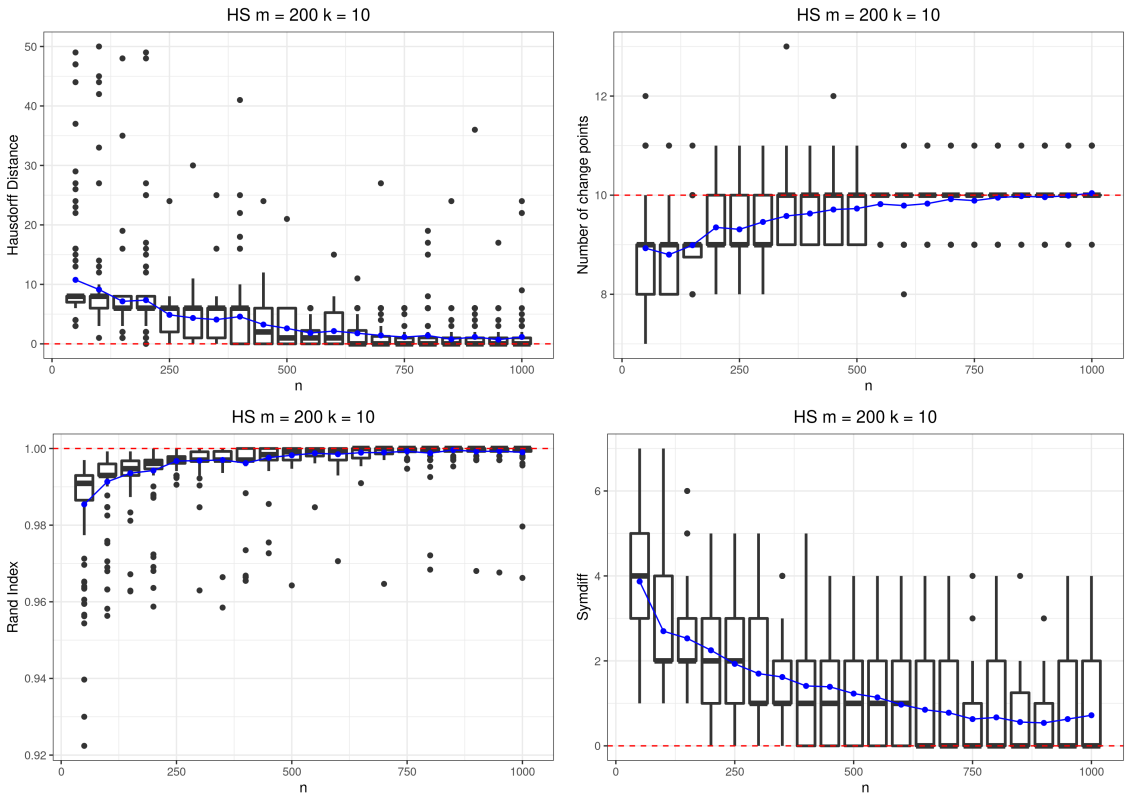


Figure E.10: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the HS*

k-10 m-500

CVSEG

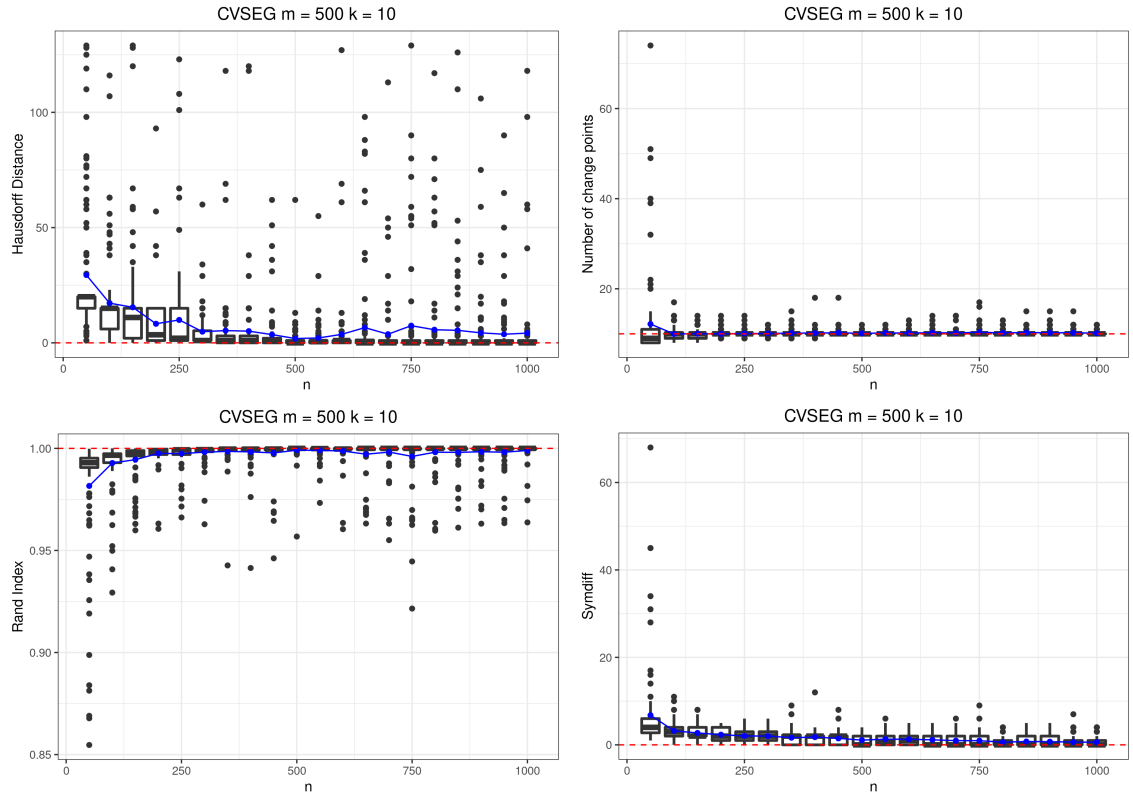


Figure E.11: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVSEG*

CVDPS

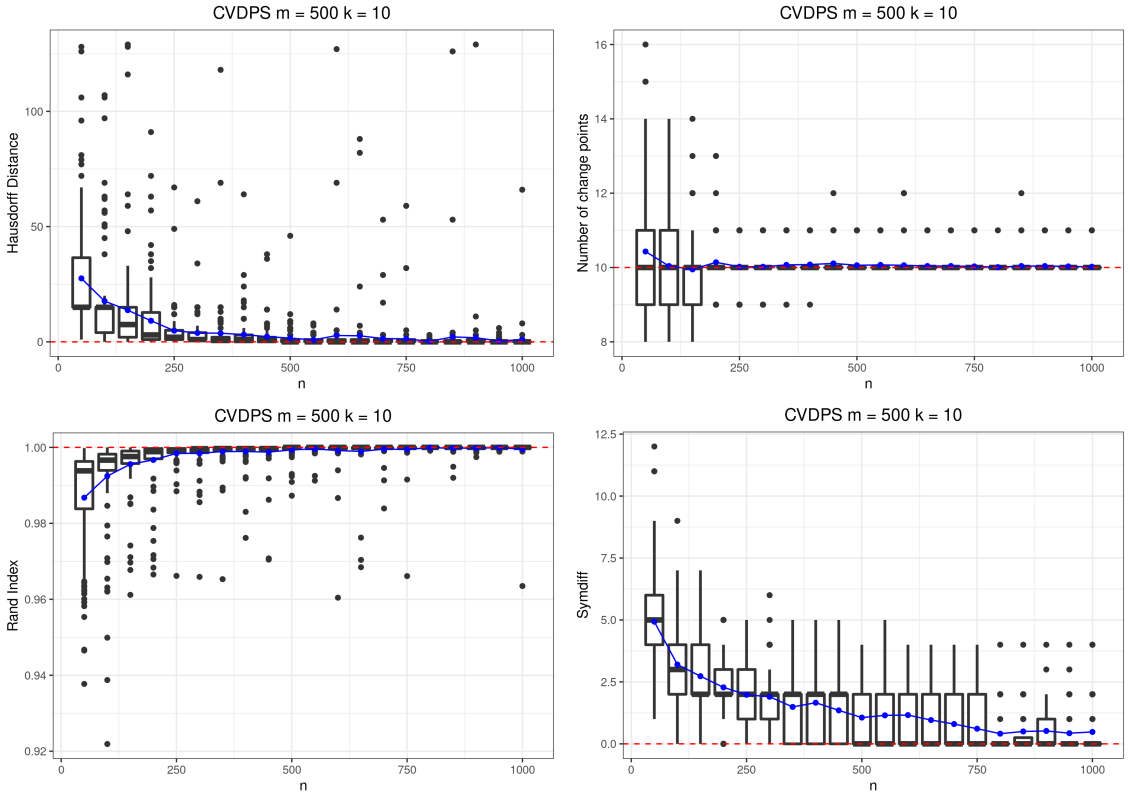


Figure E.12: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVDPS*

DPS LOG

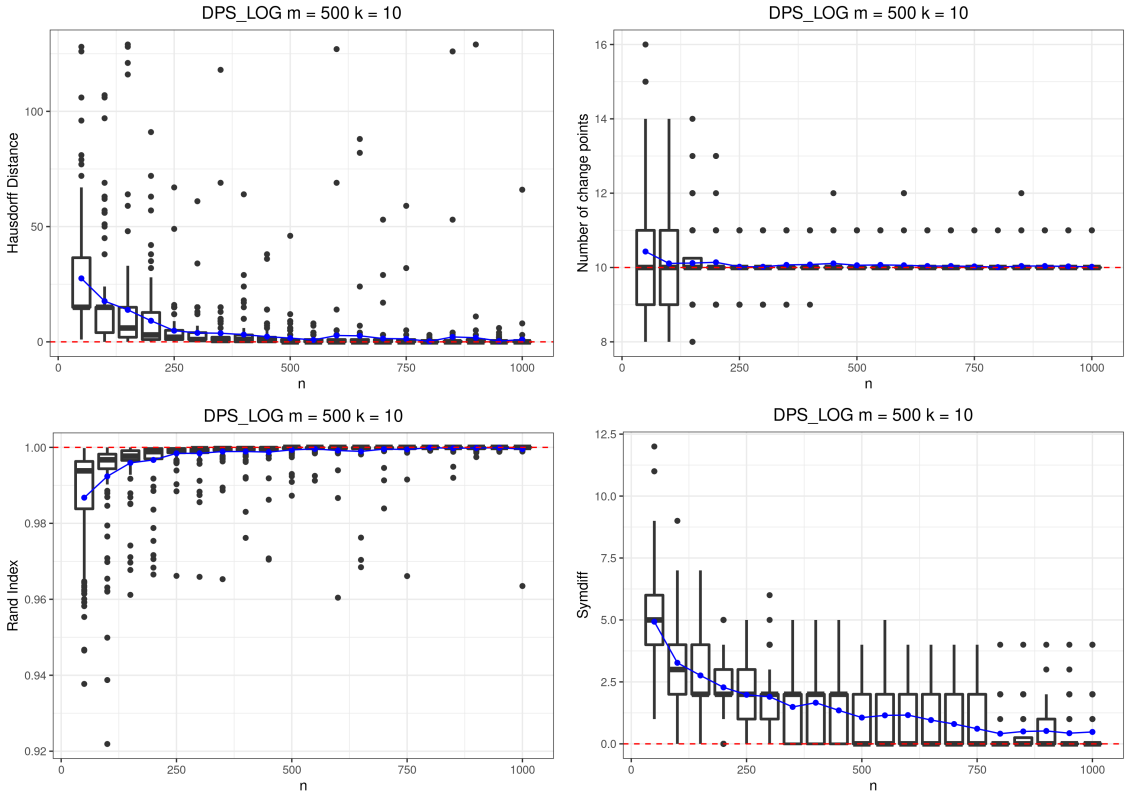


Figure E.13: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS LOG*

DPS SQRT

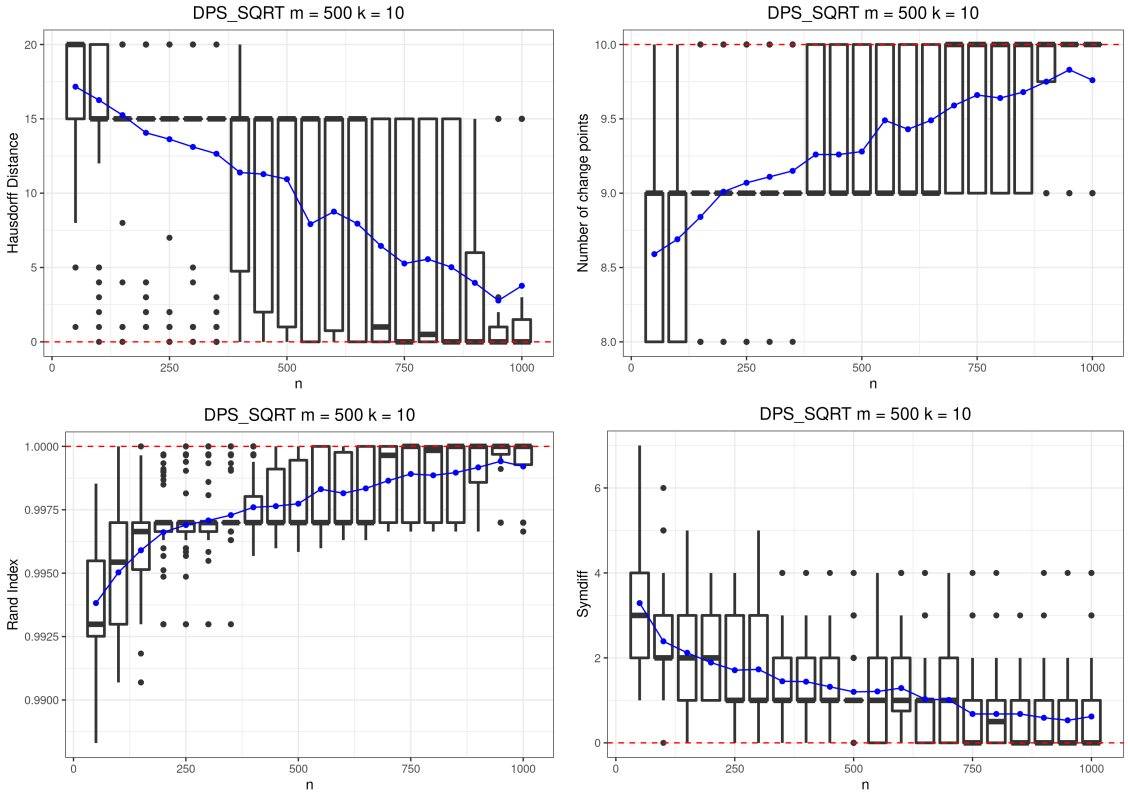


Figure E.14: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS SQRT*

HS

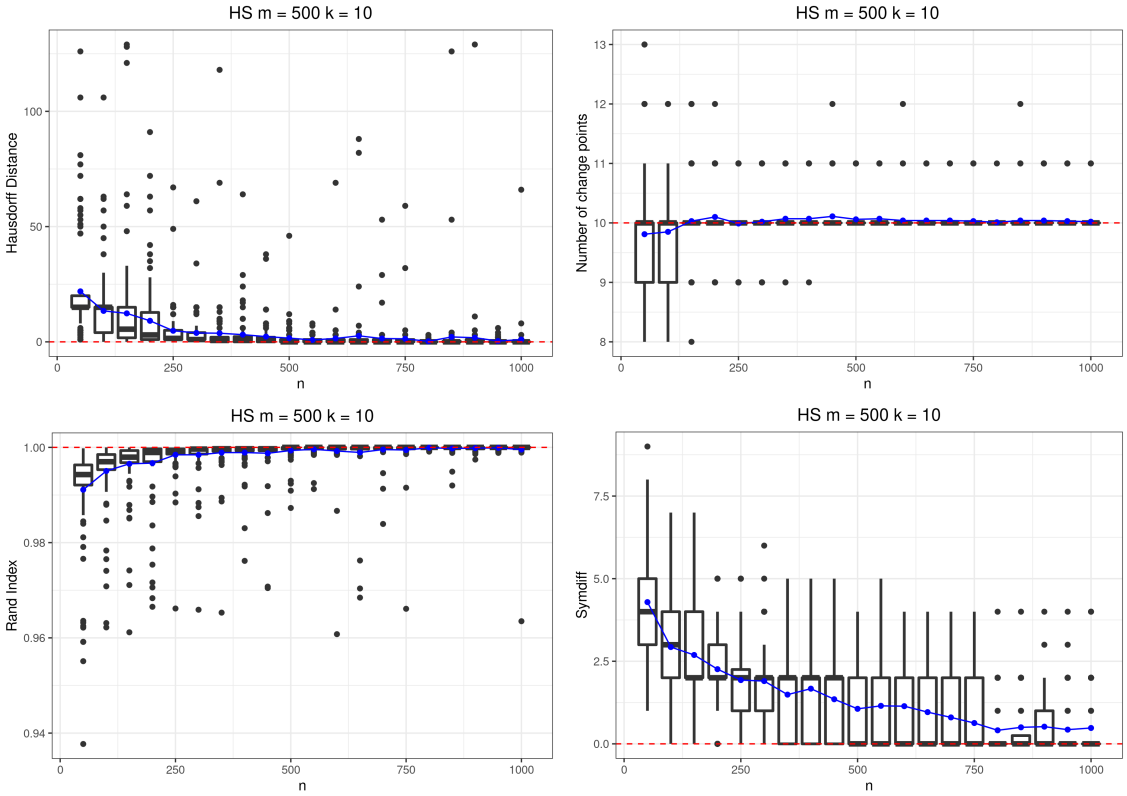


Figure E.15: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the HS*

k-50 m-100

CVSEG

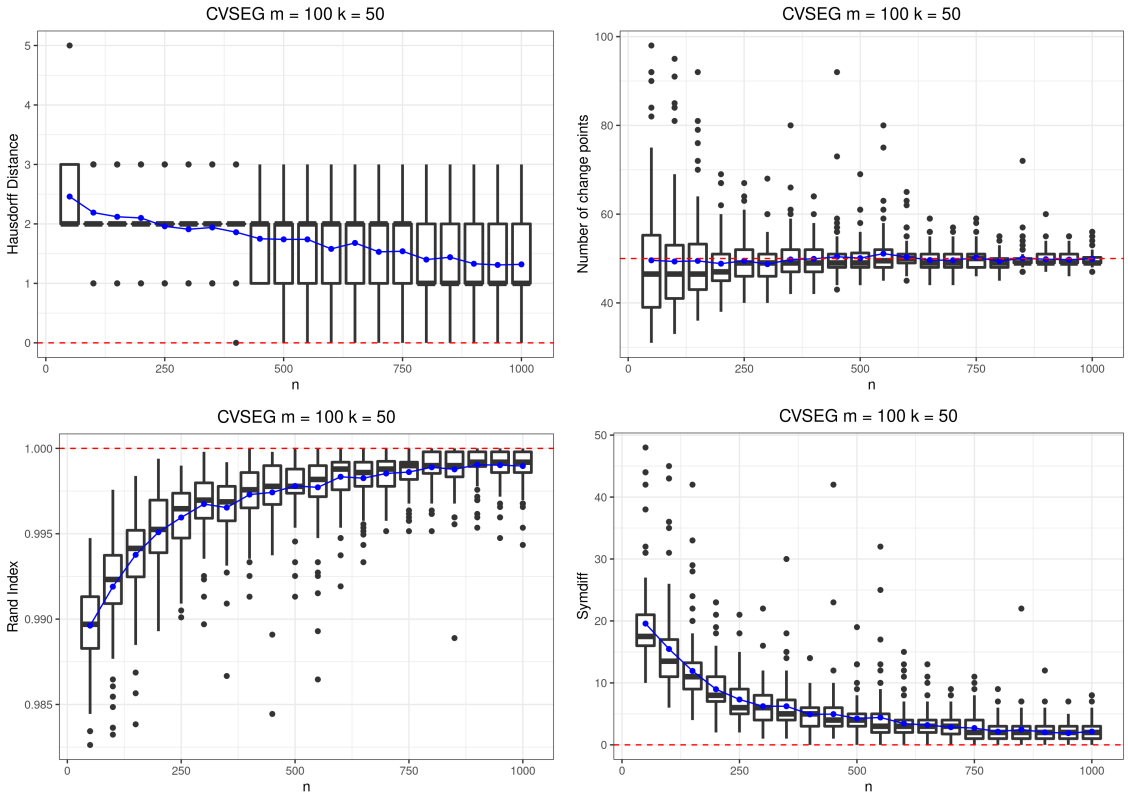


Figure E.16: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVSEG*

CVDPS

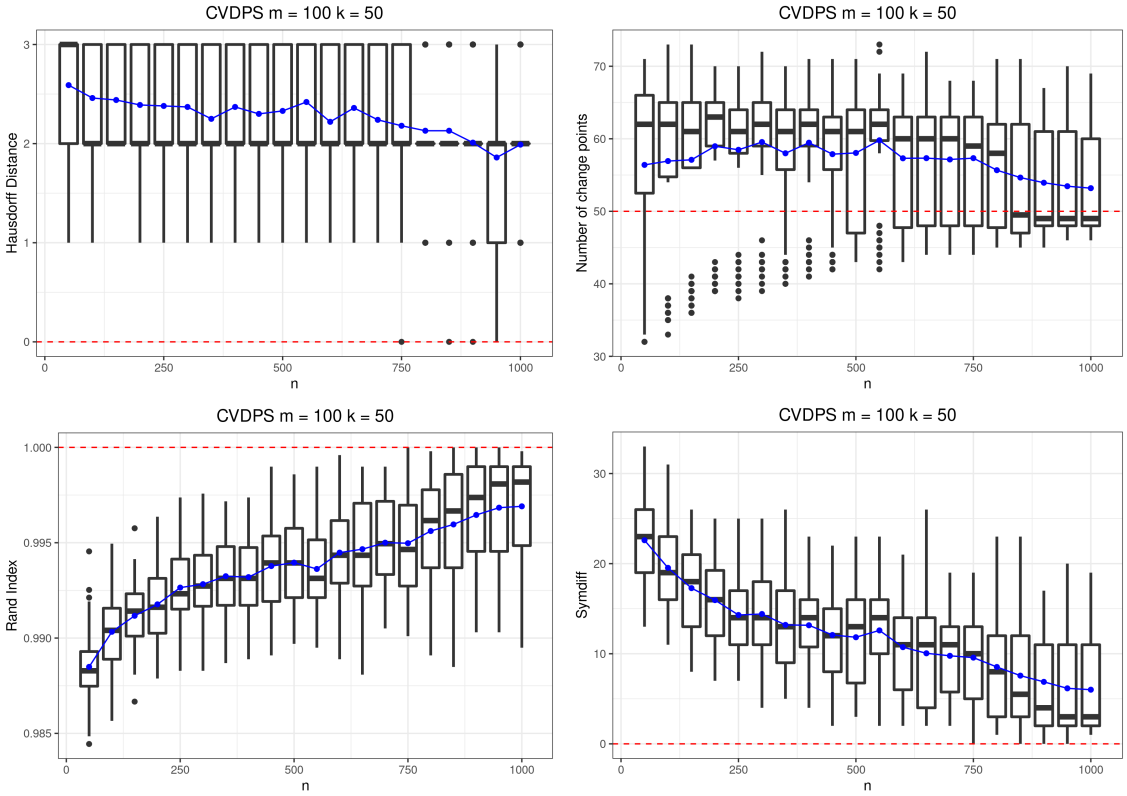


Figure E.17: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVDPS*

DPS LOG

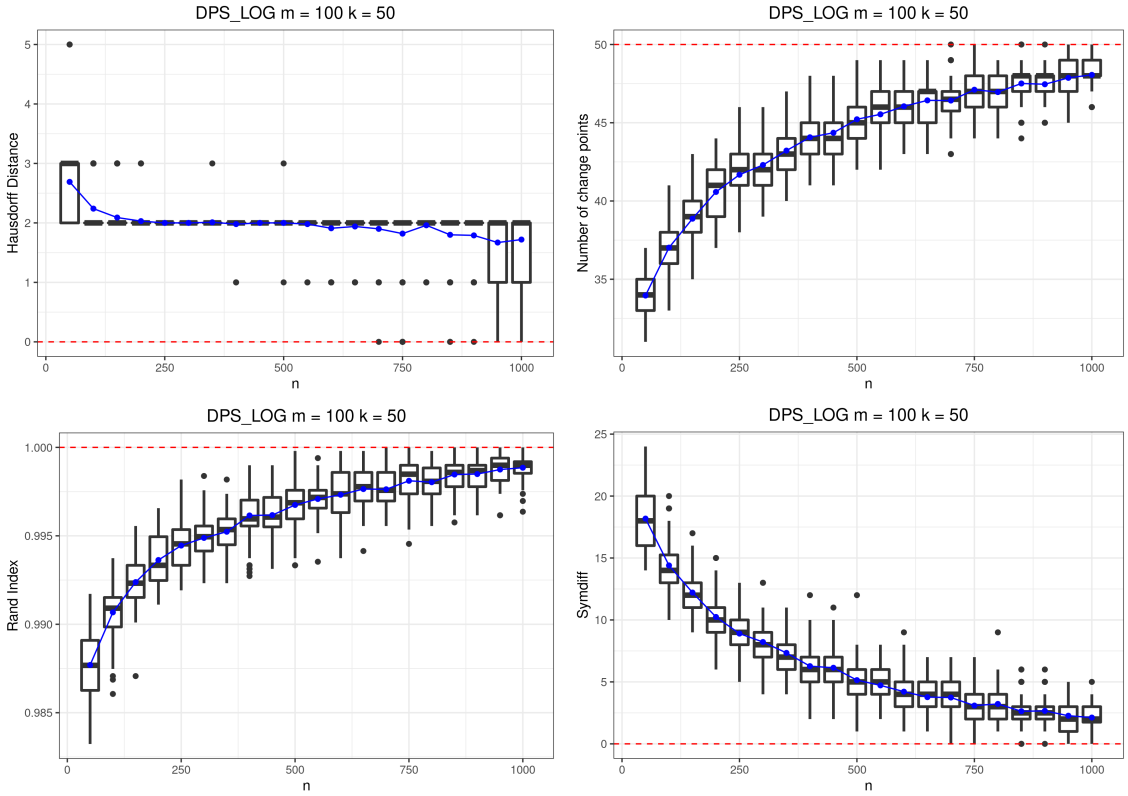


Figure E.18: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS LOG*

DPS SQRT

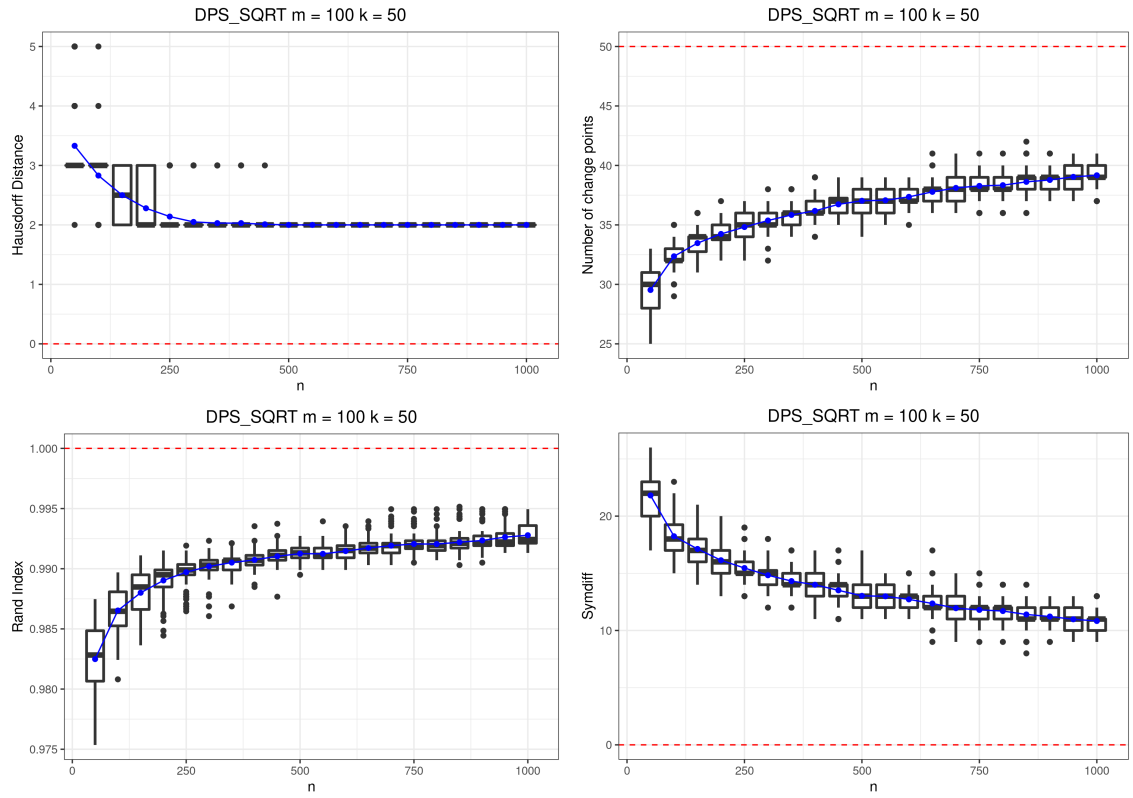


Figure E.19: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS SQRT*

HS

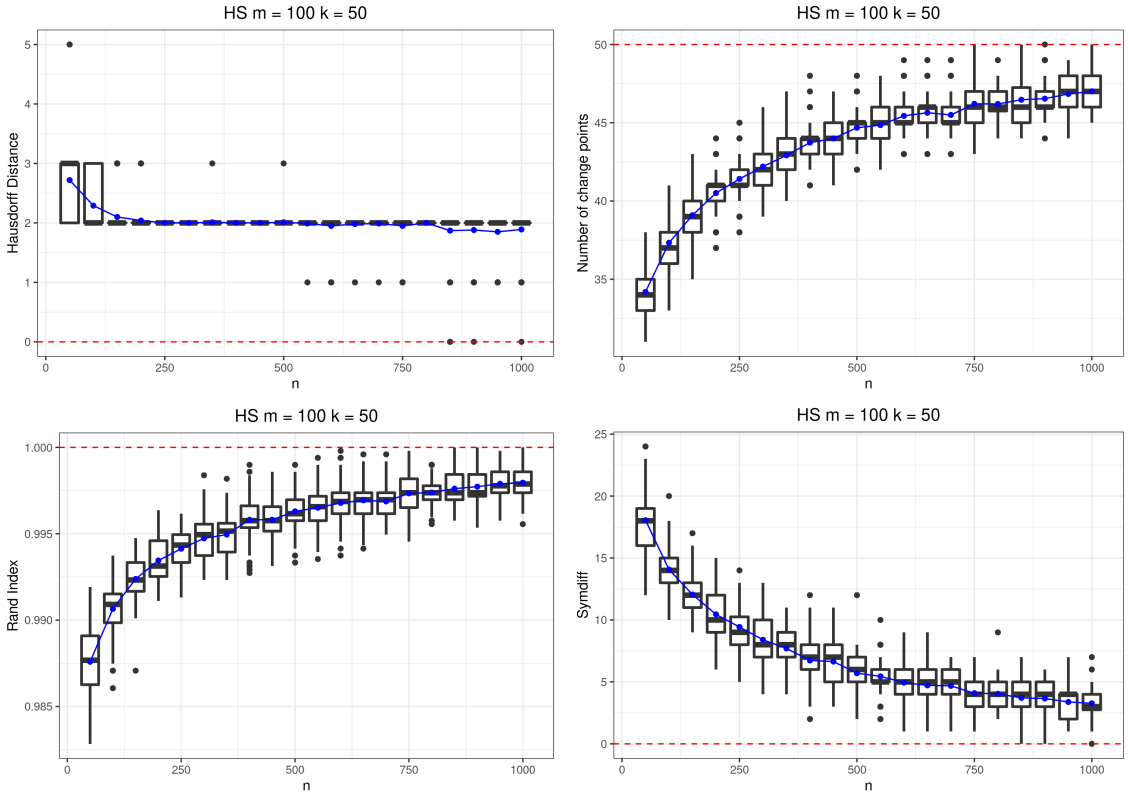


Figure E.20: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the HS*

k-50 m-200

CVSEG

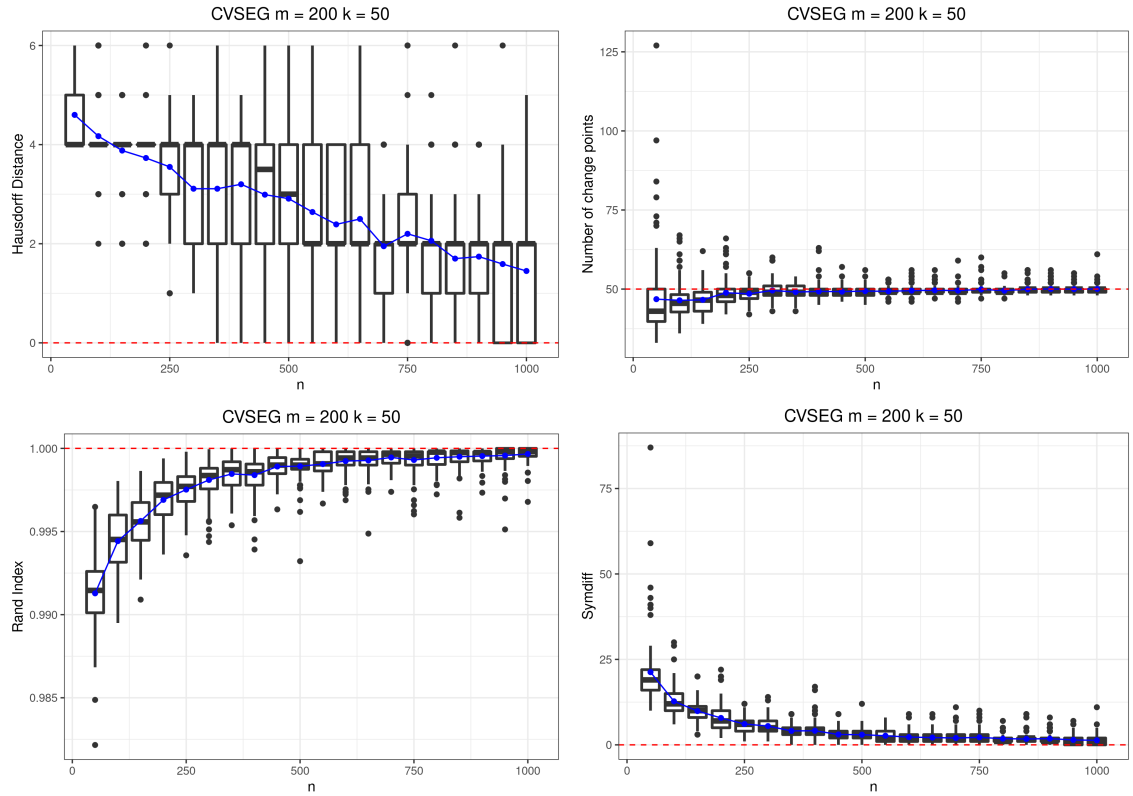


Figure E.21: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVSEG*

CVDPS

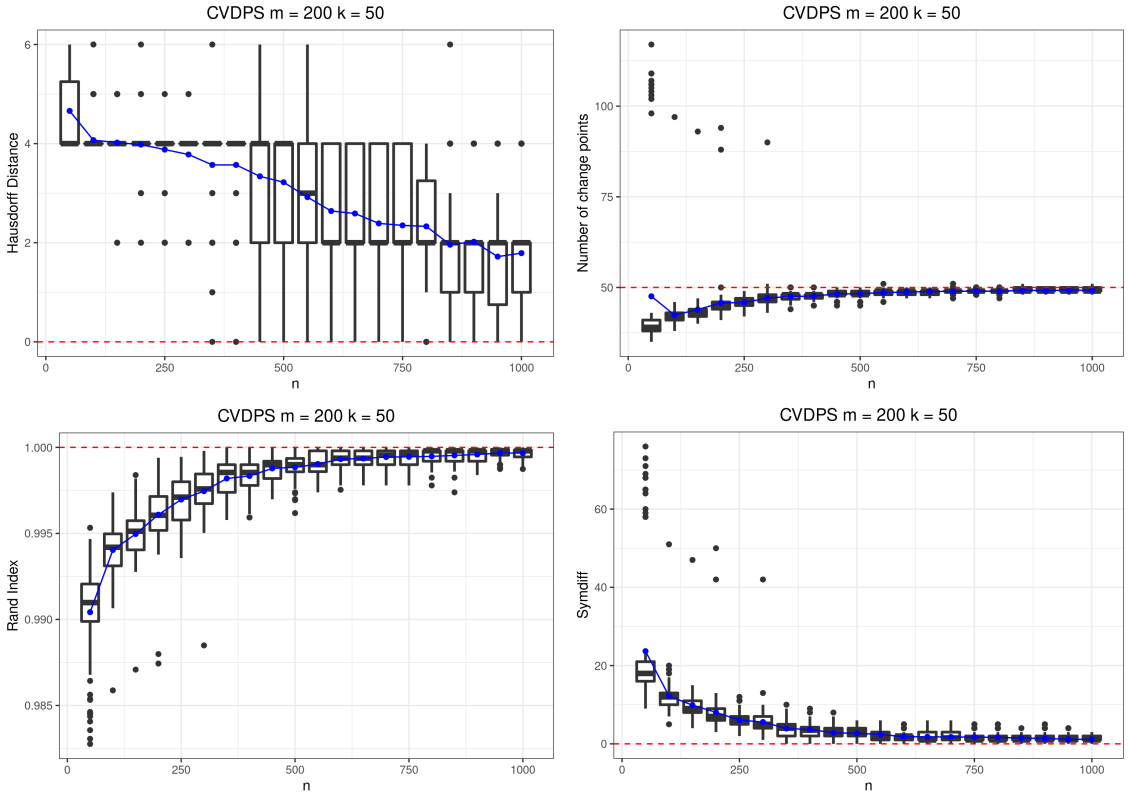


Figure E.22: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVDPS*

DPS LOG

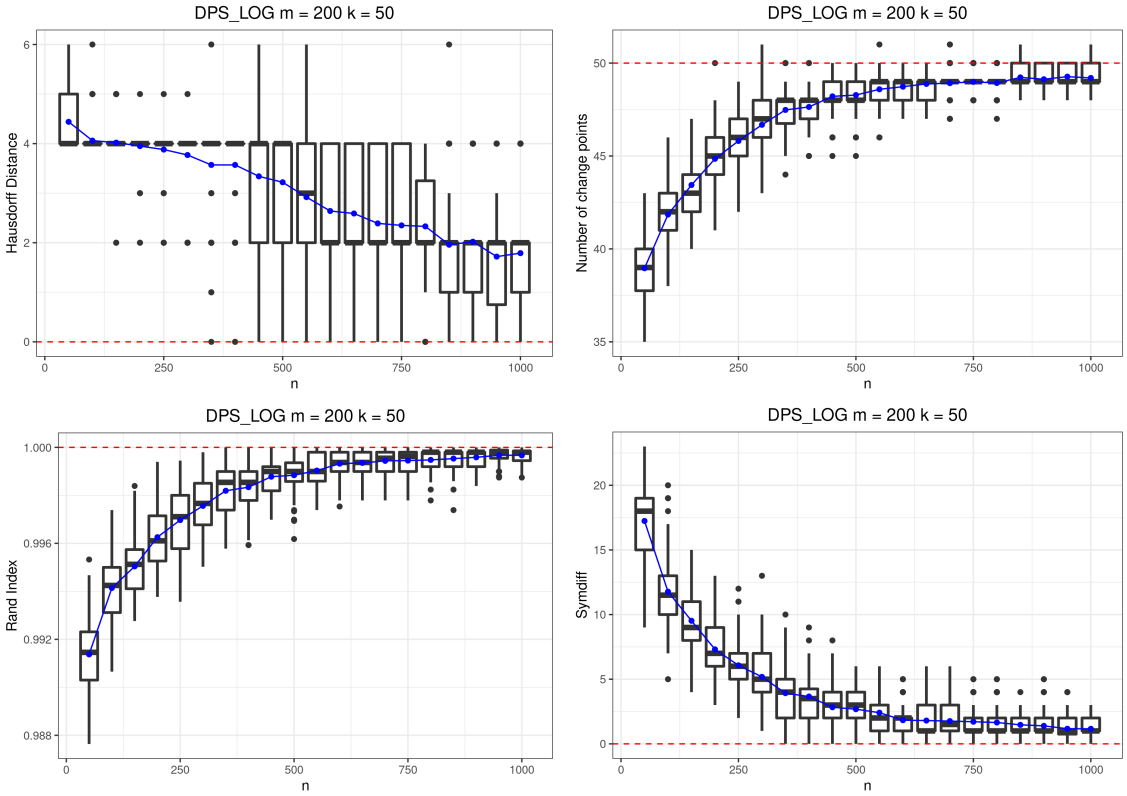


Figure E.23: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS LOG*

DPS SQRT

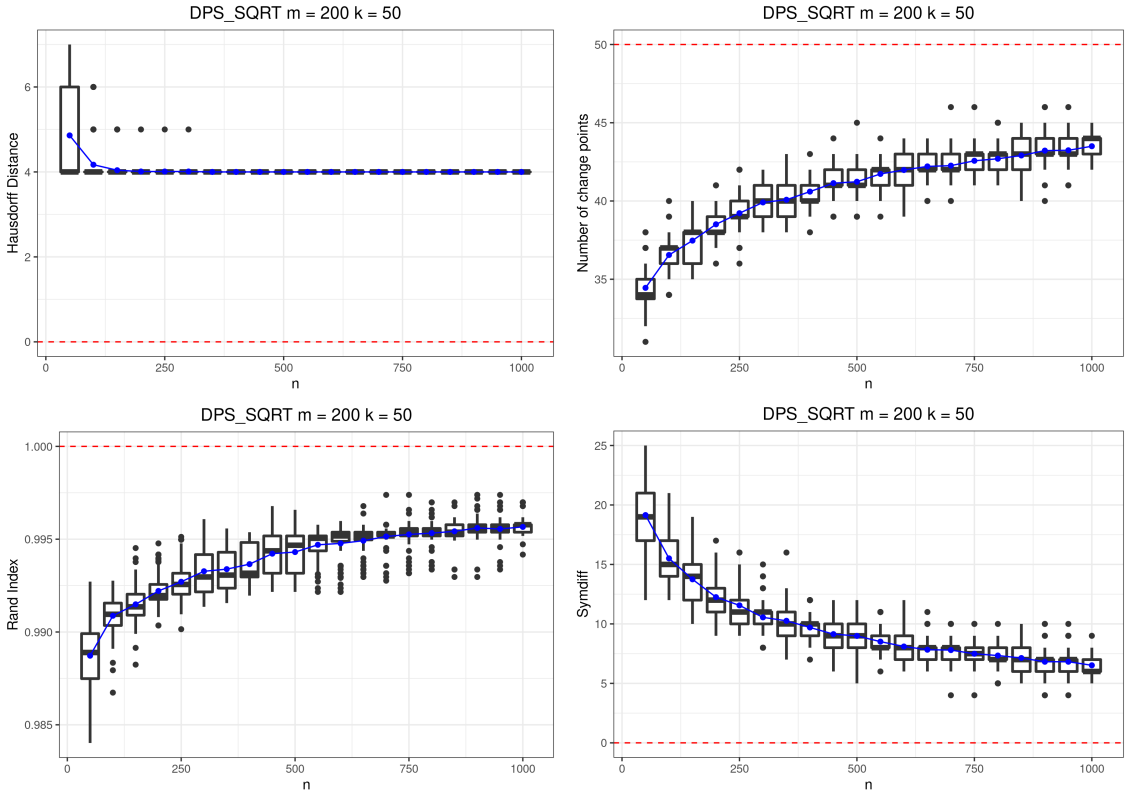


Figure E.24: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS SQRT*

HS

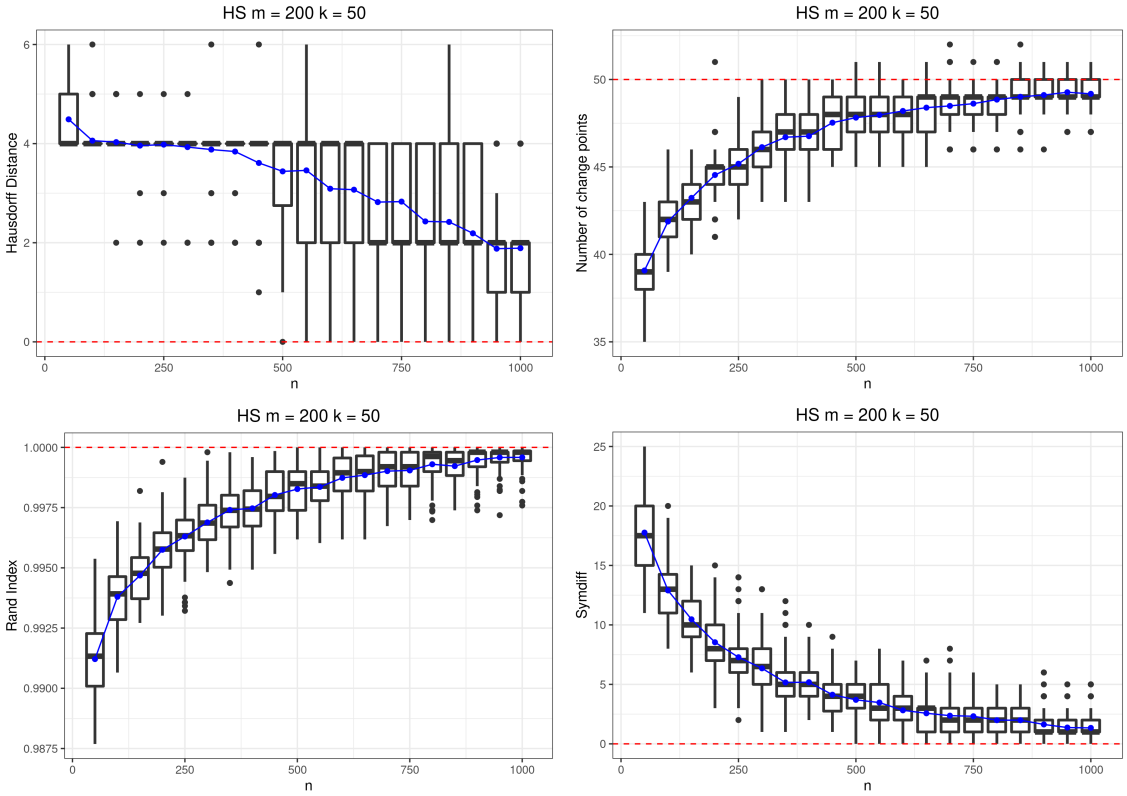


Figure E.25: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the HS*

k-50 m-500

CVSEG

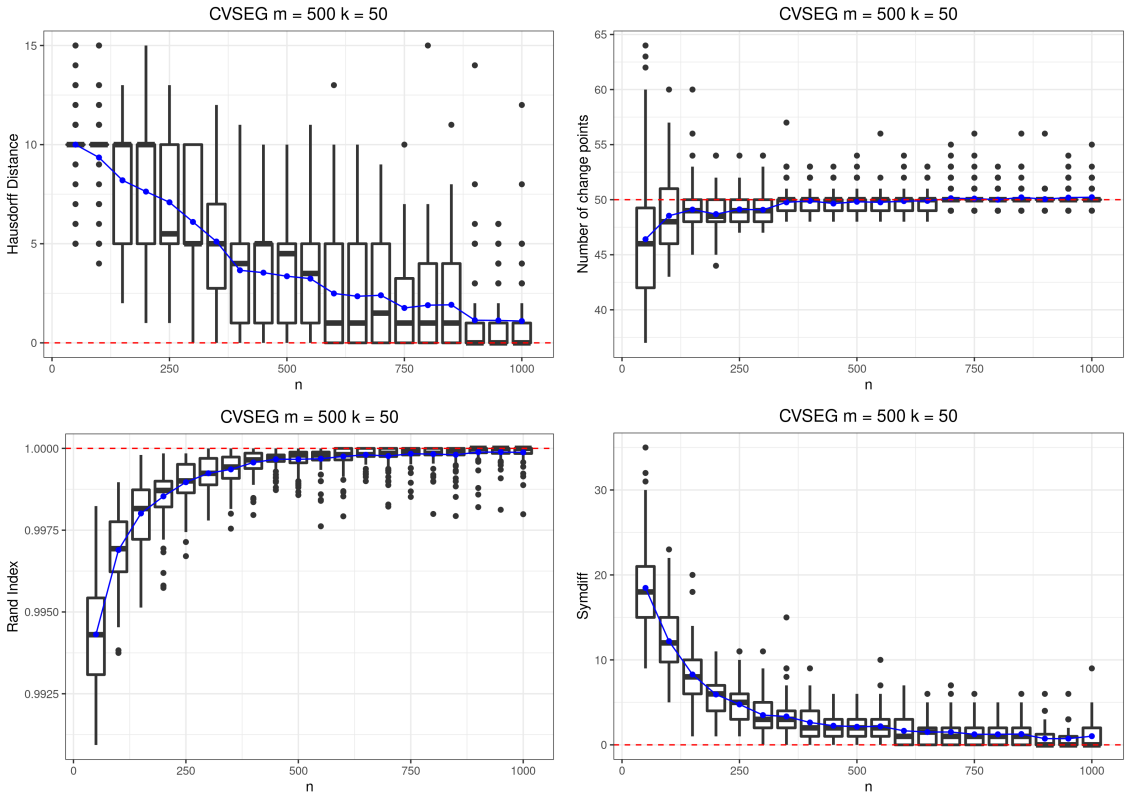


Figure E.26: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVSEG*

CVDPS

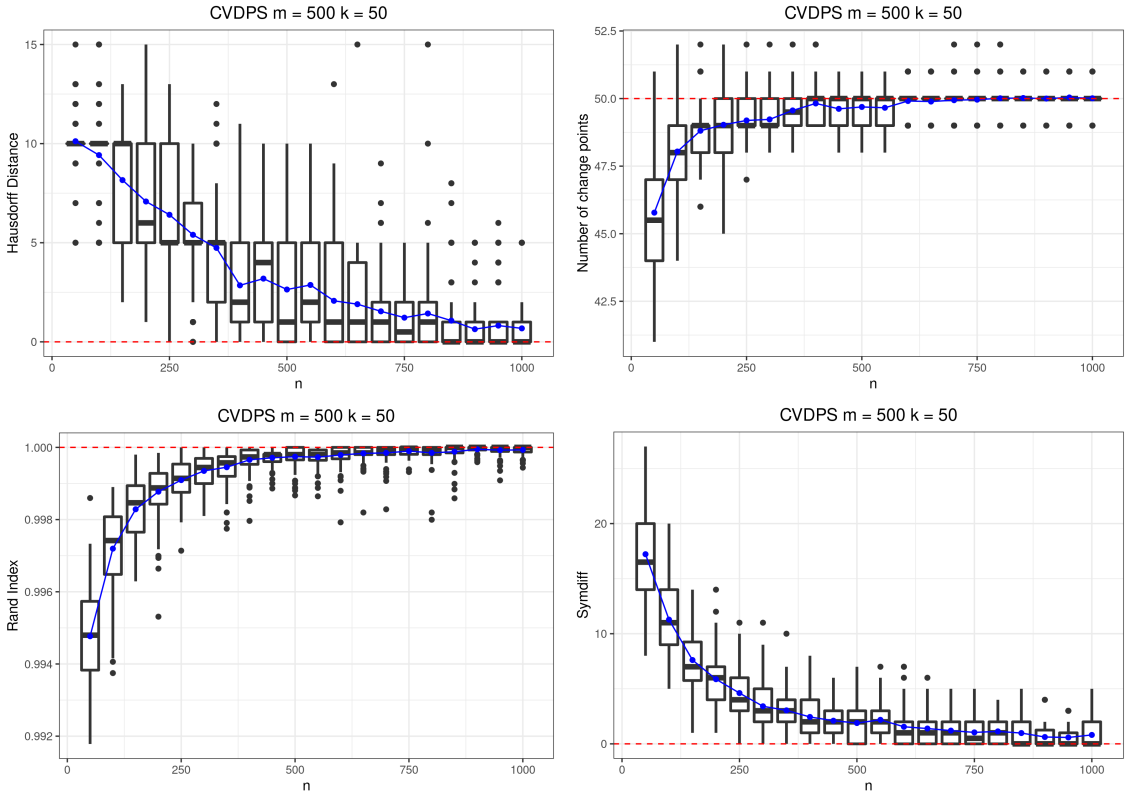


Figure E.27: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVDPS*

DPS LOG

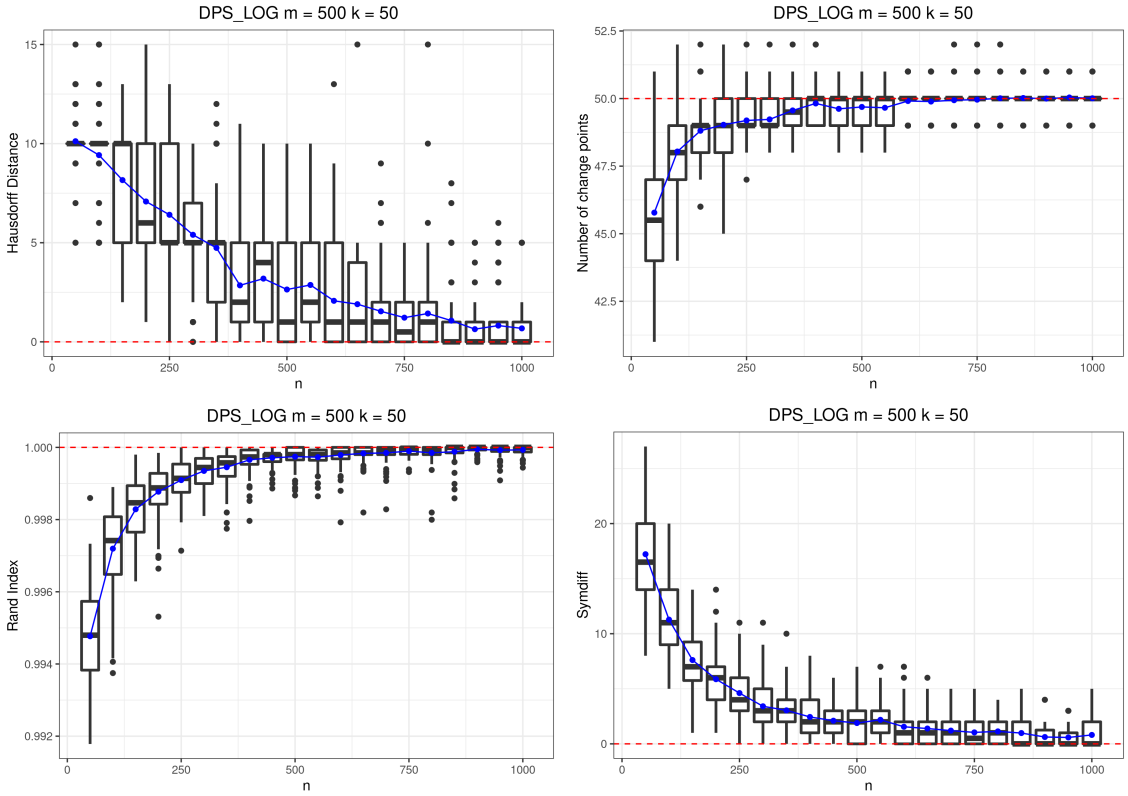


Figure E.28: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS LOG*

DPS SQRT

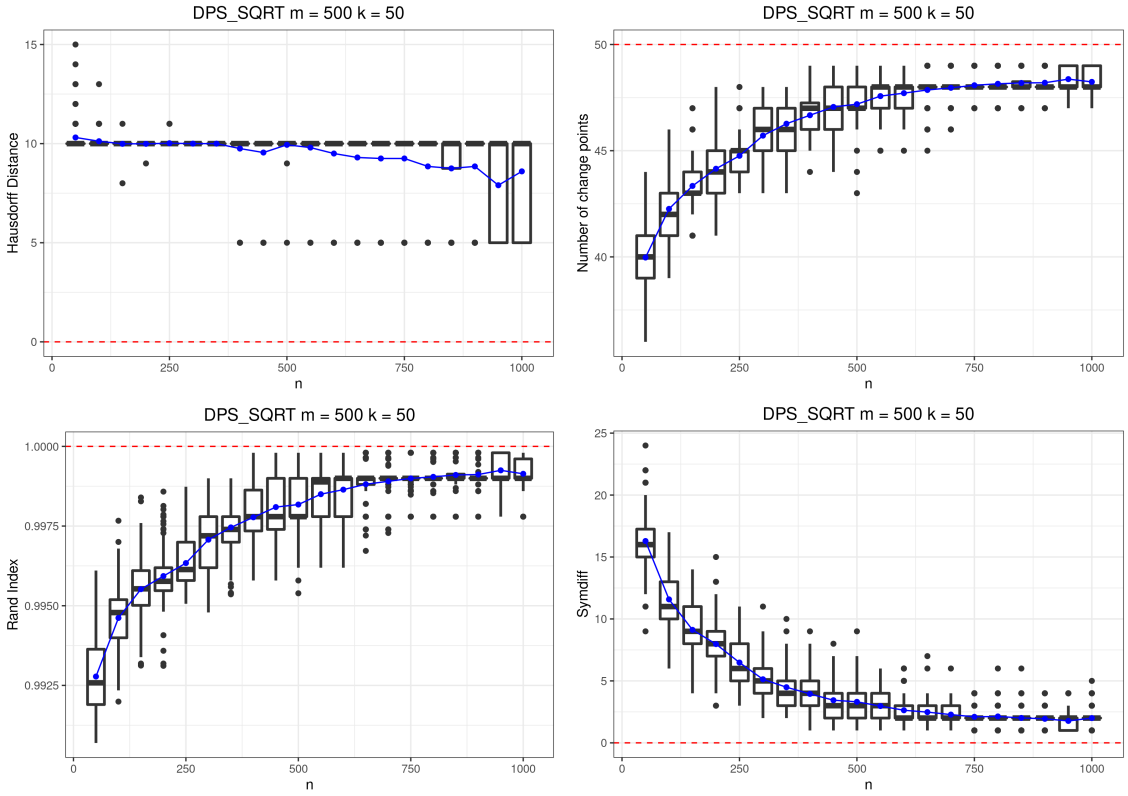


Figure E.29: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS SQRT*

HS

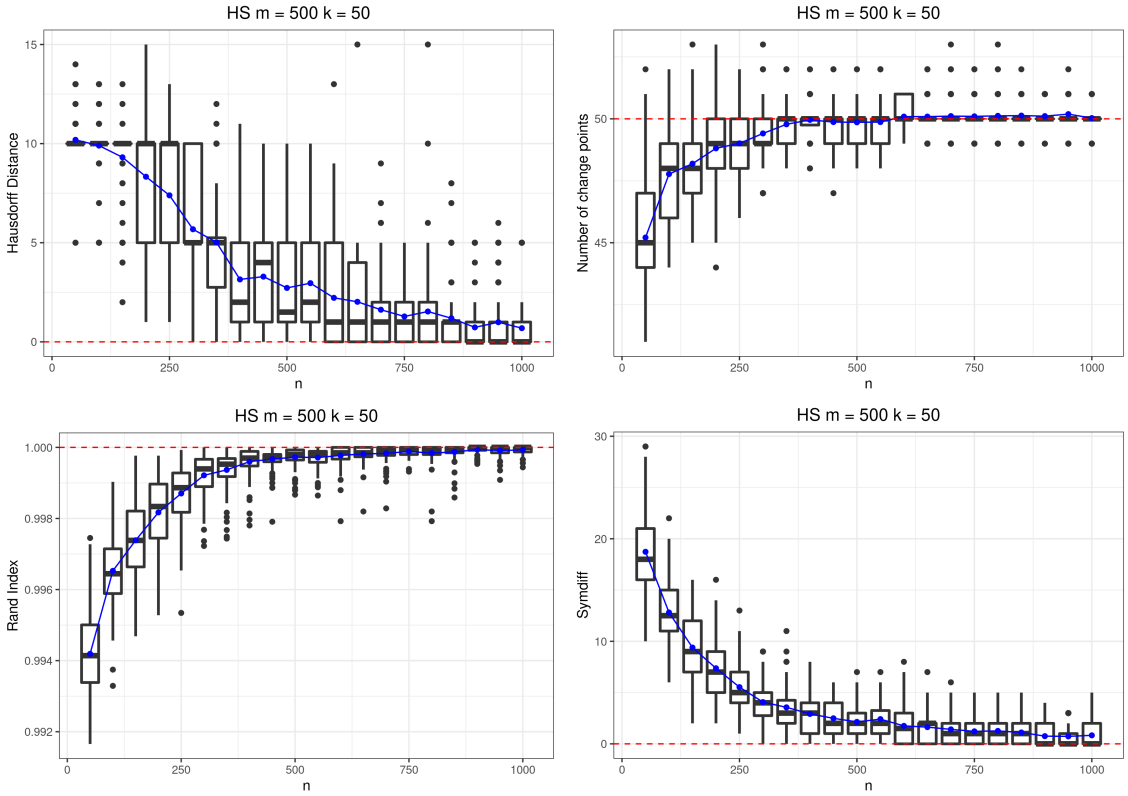


Figure E.30: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the HS*

k-90 m-100

CVSEG

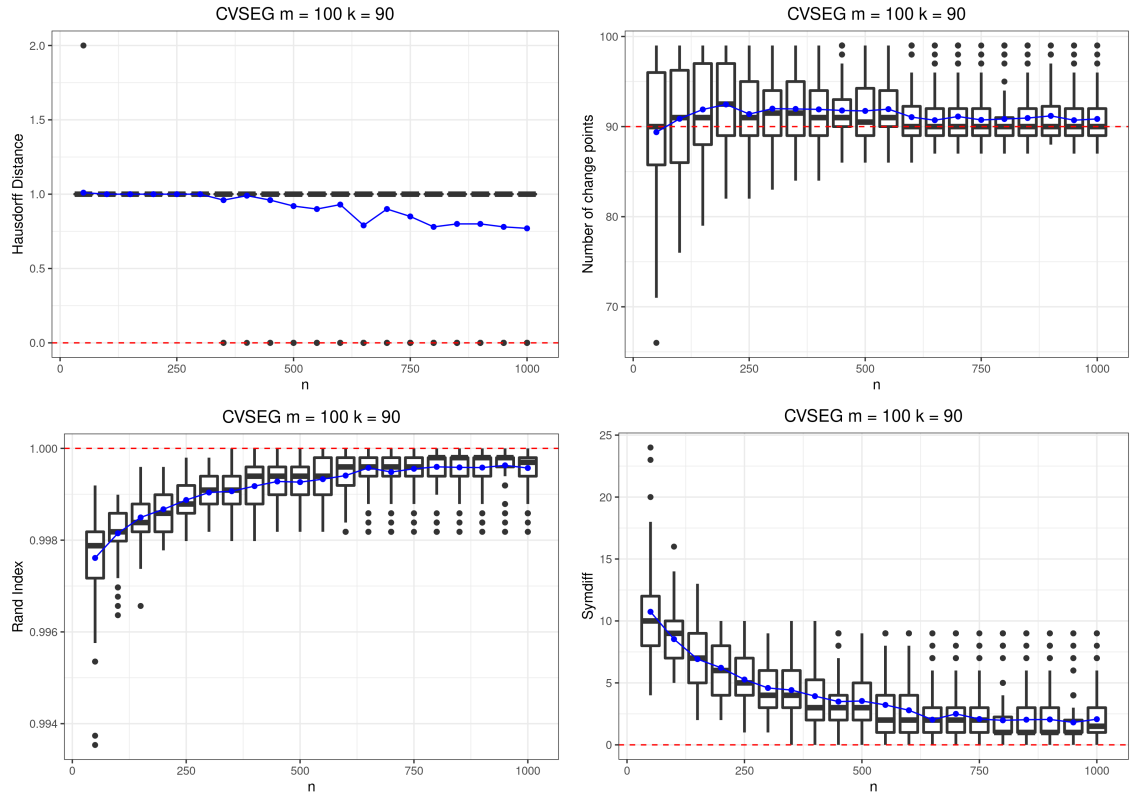


Figure E.31: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVSEG*

CVDPS

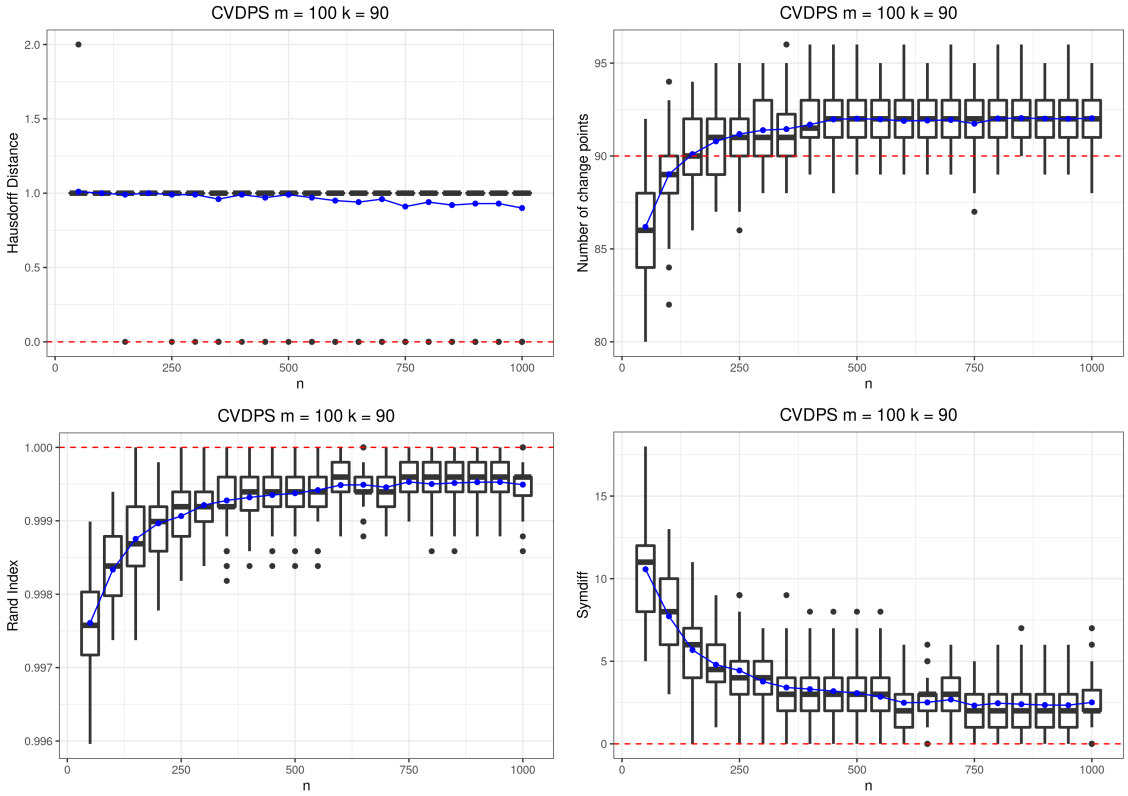


Figure E.32: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVDPS*

DPS LOG

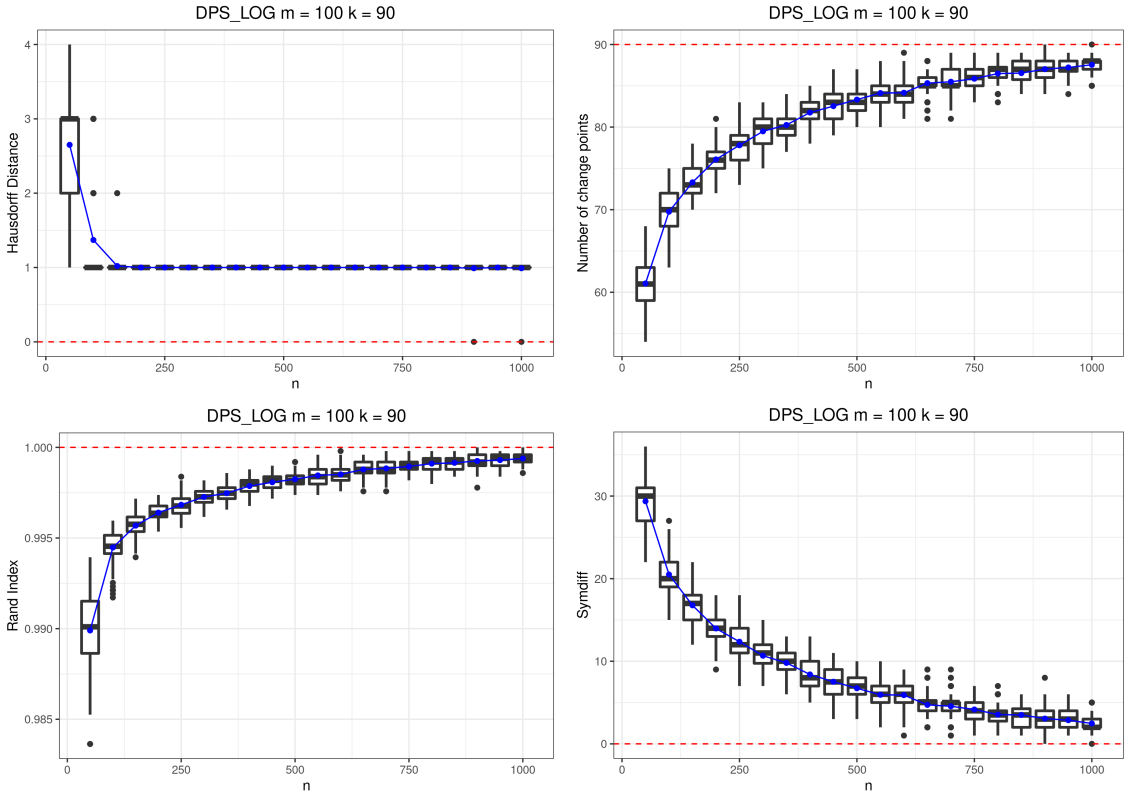


Figure E.33: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS LOG*

DPS SQRT

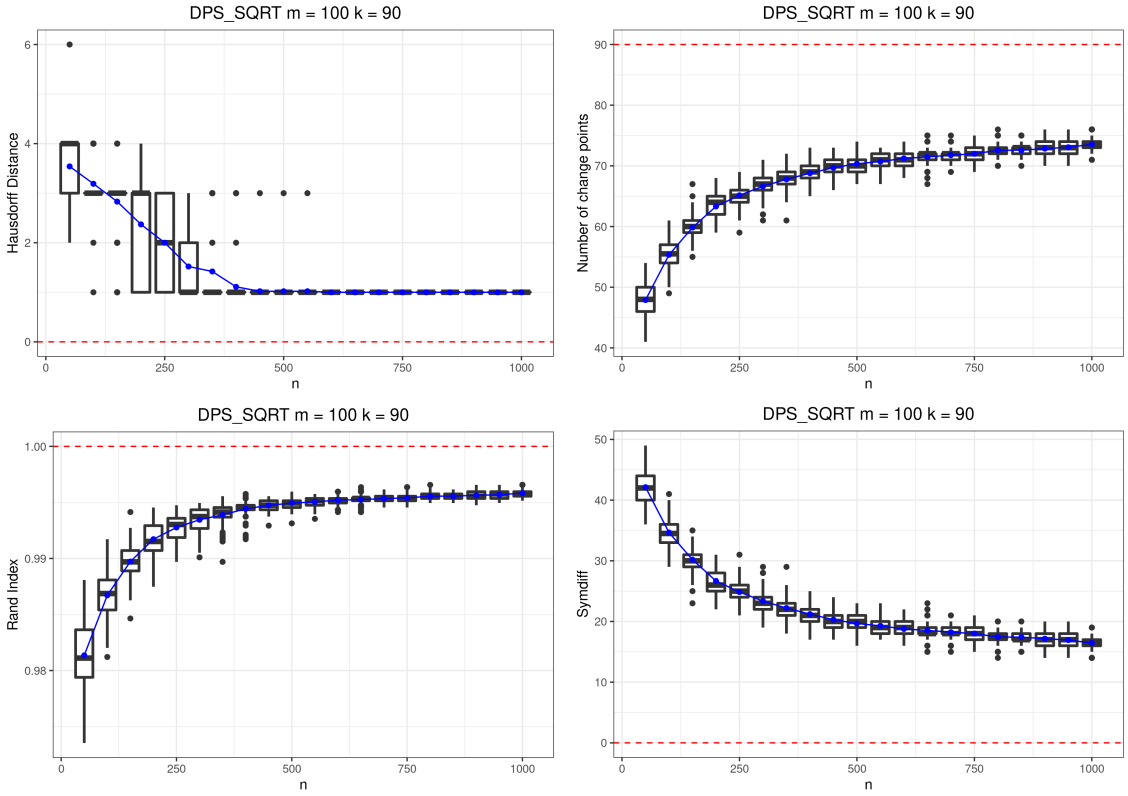


Figure E.34: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS SQRT*

HS

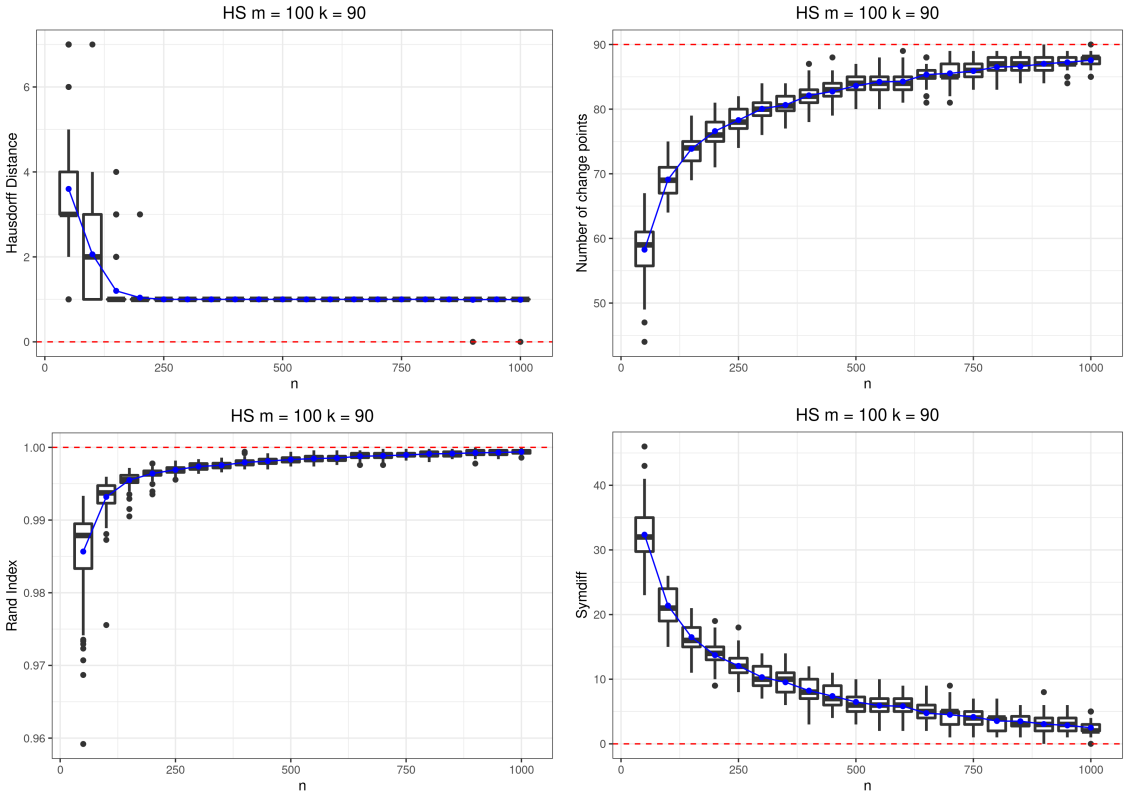


Figure E.35: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the HS*

k-90 m-200

CVSEG

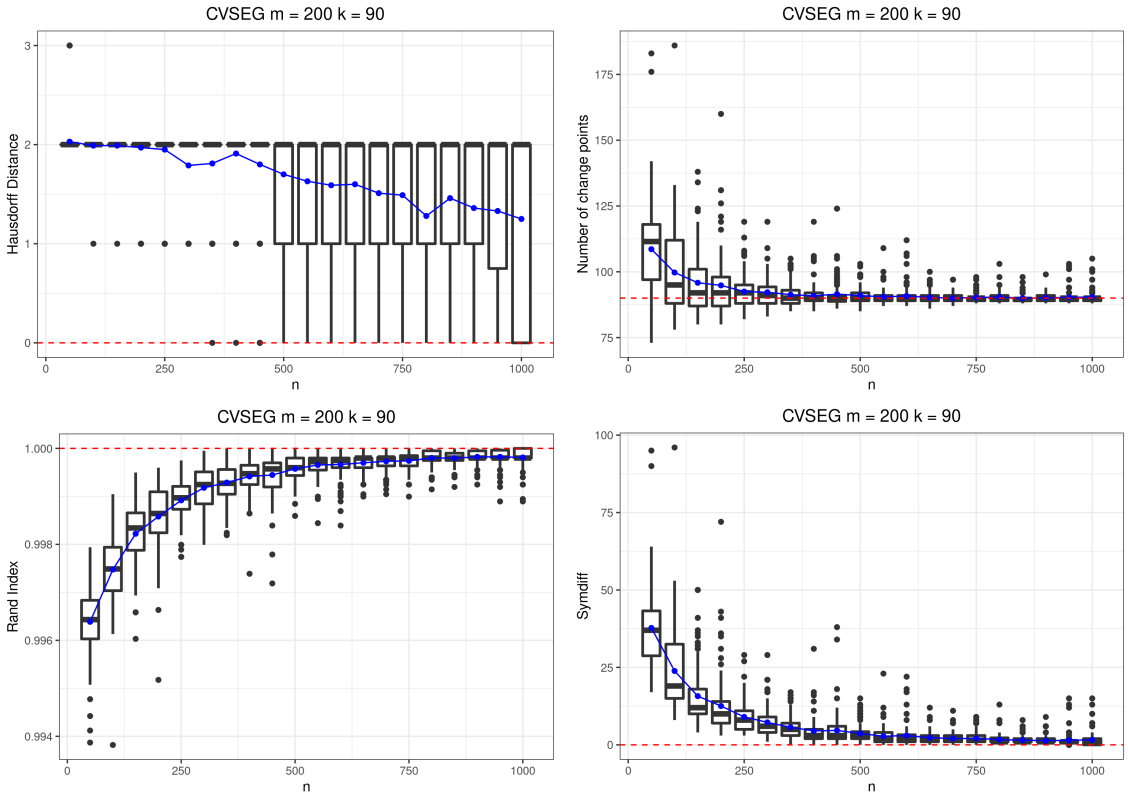


Figure E.36: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVSEG*

CVDPS

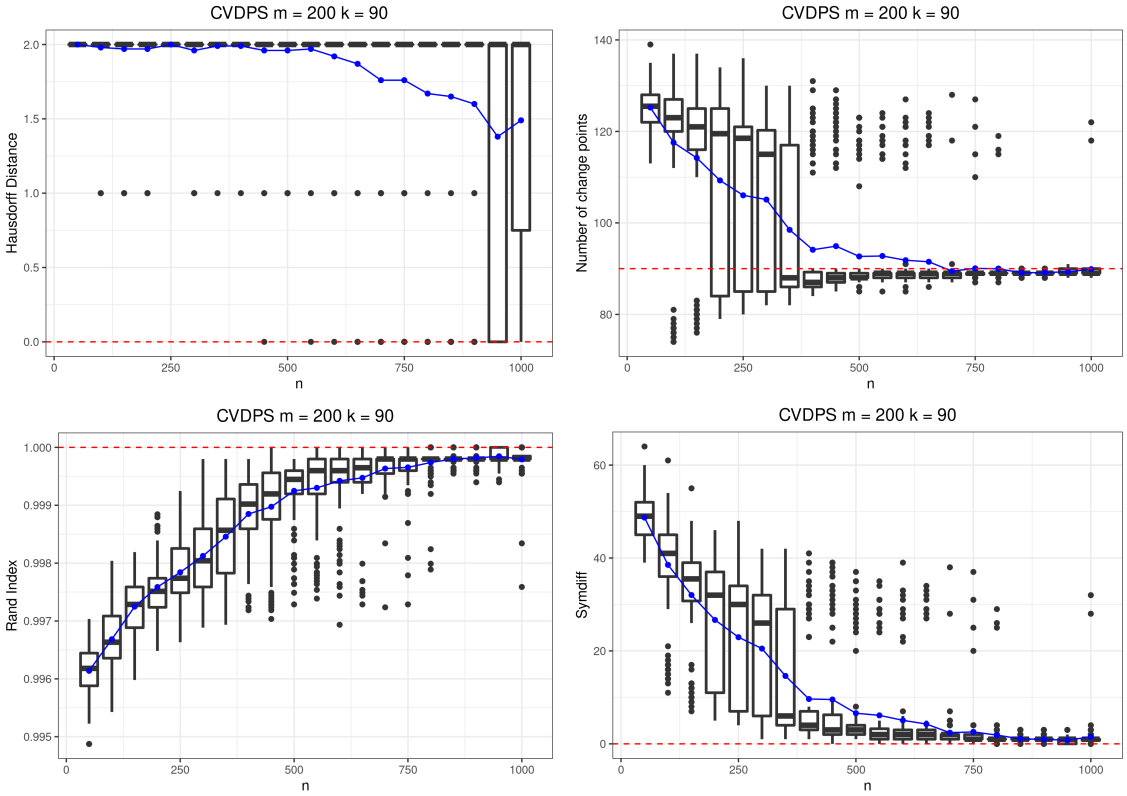


Figure E.37: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVDPS*

DPS LOG

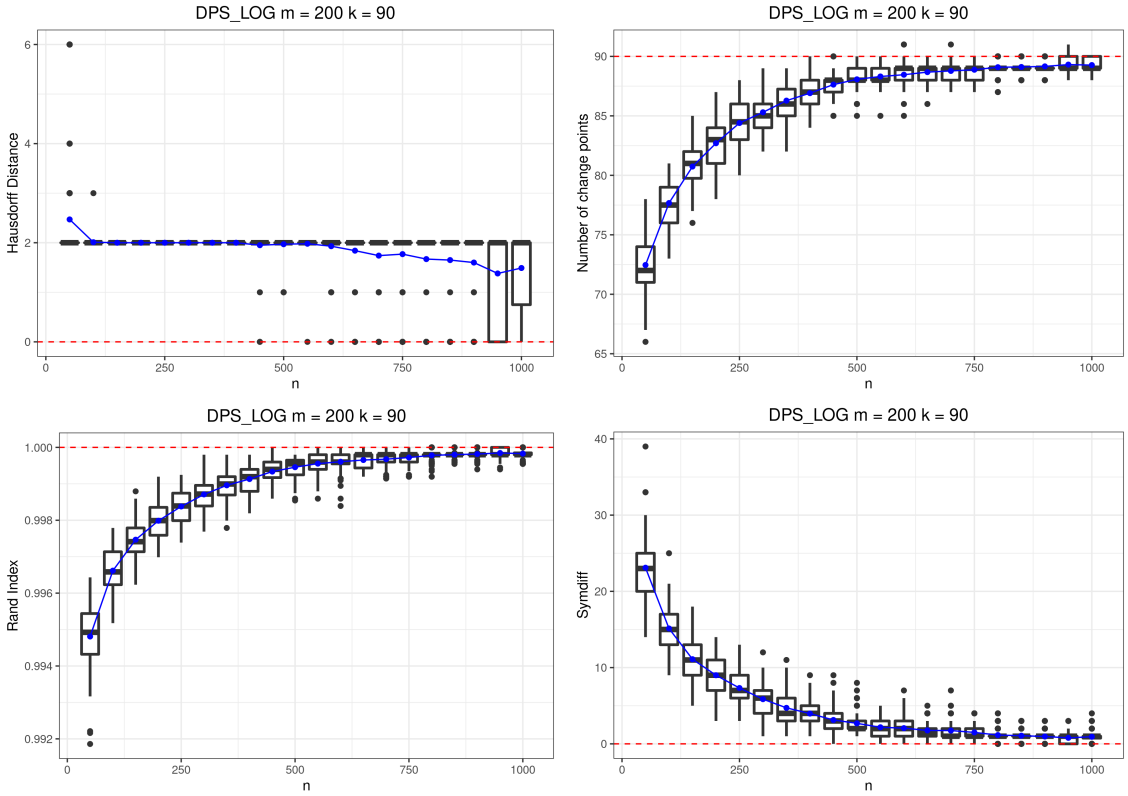


Figure E.38: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS LOG*

DPS SQRT

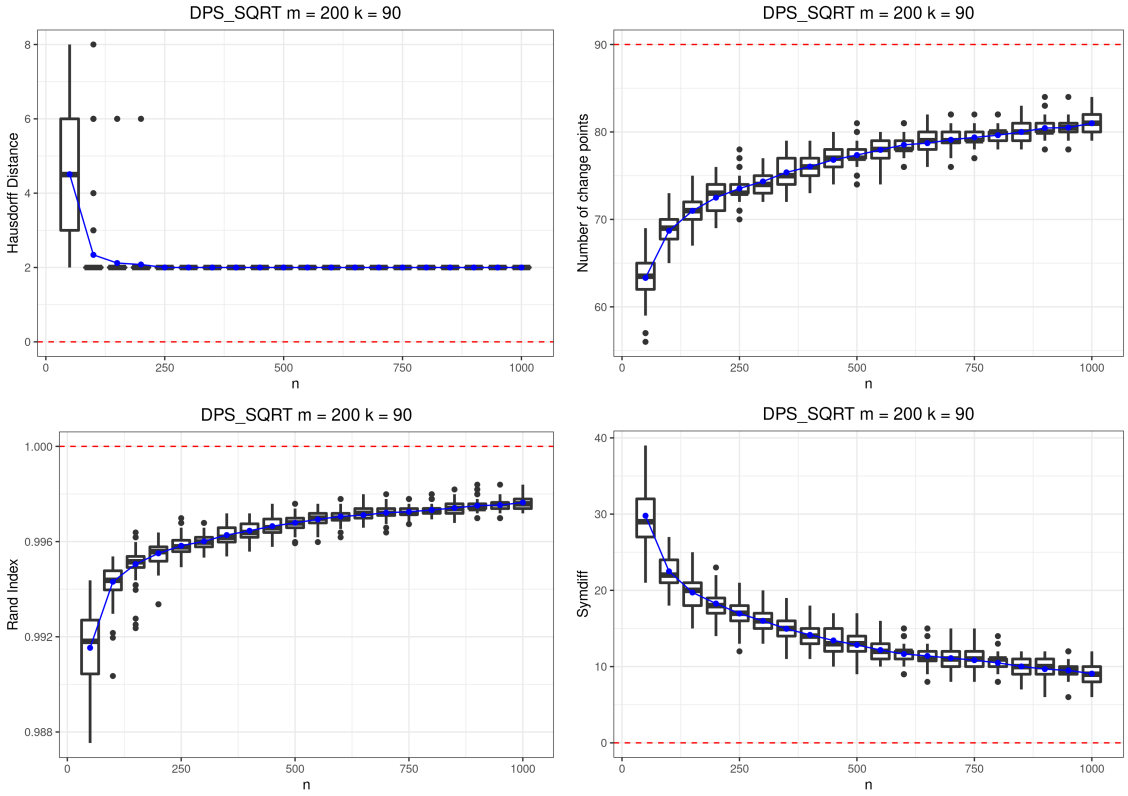


Figure E.39: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS SQRT*

HS

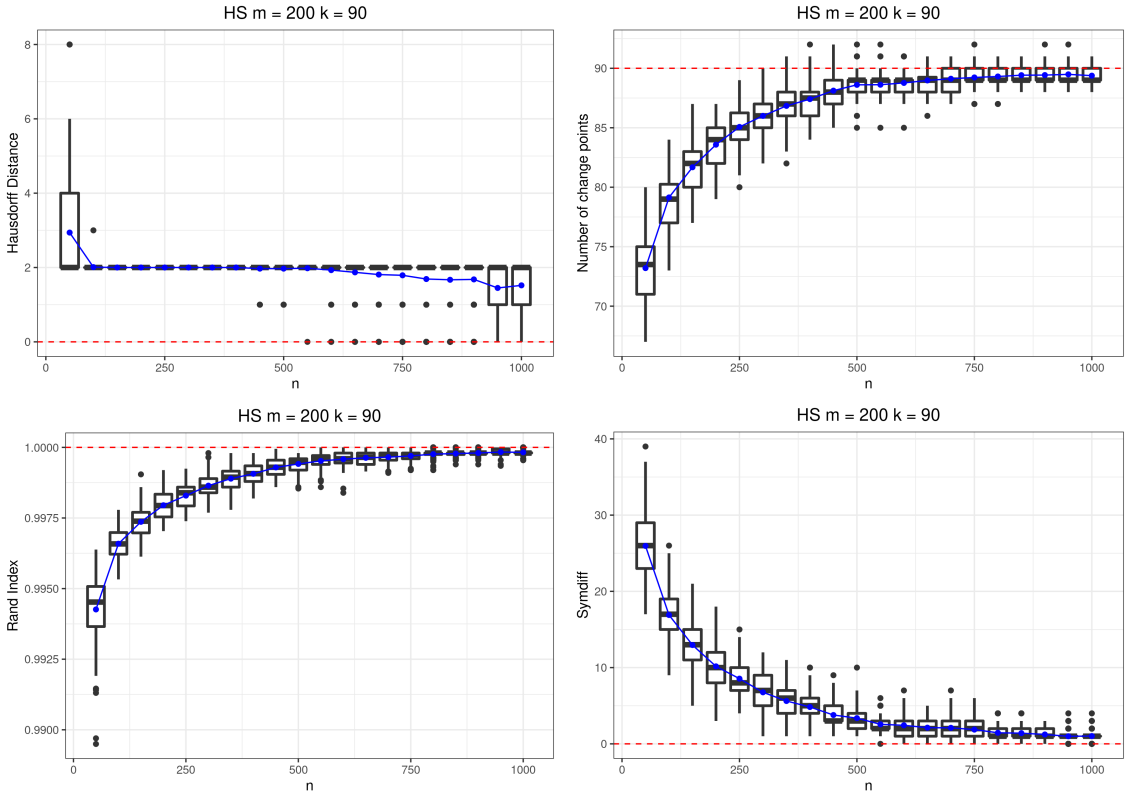


Figure E.40: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the HS*

k-90 m-500

CVSEG

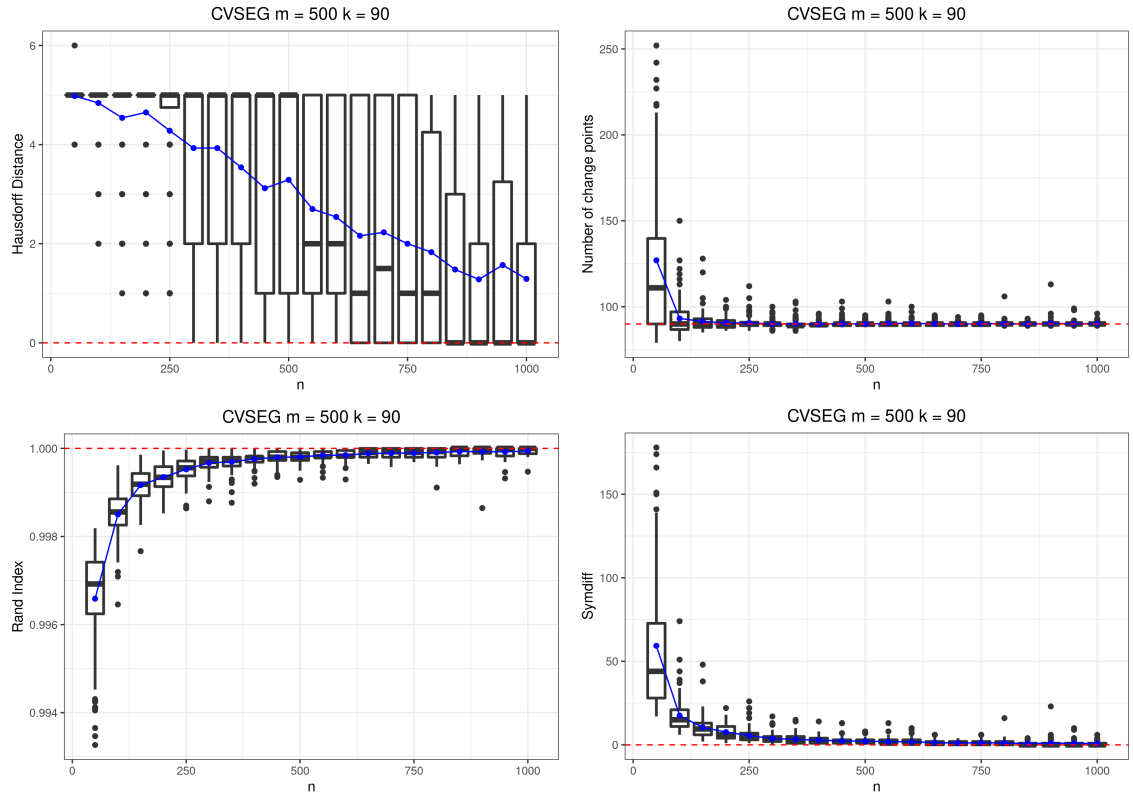


Figure E.41: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVSEG*

CVDPS

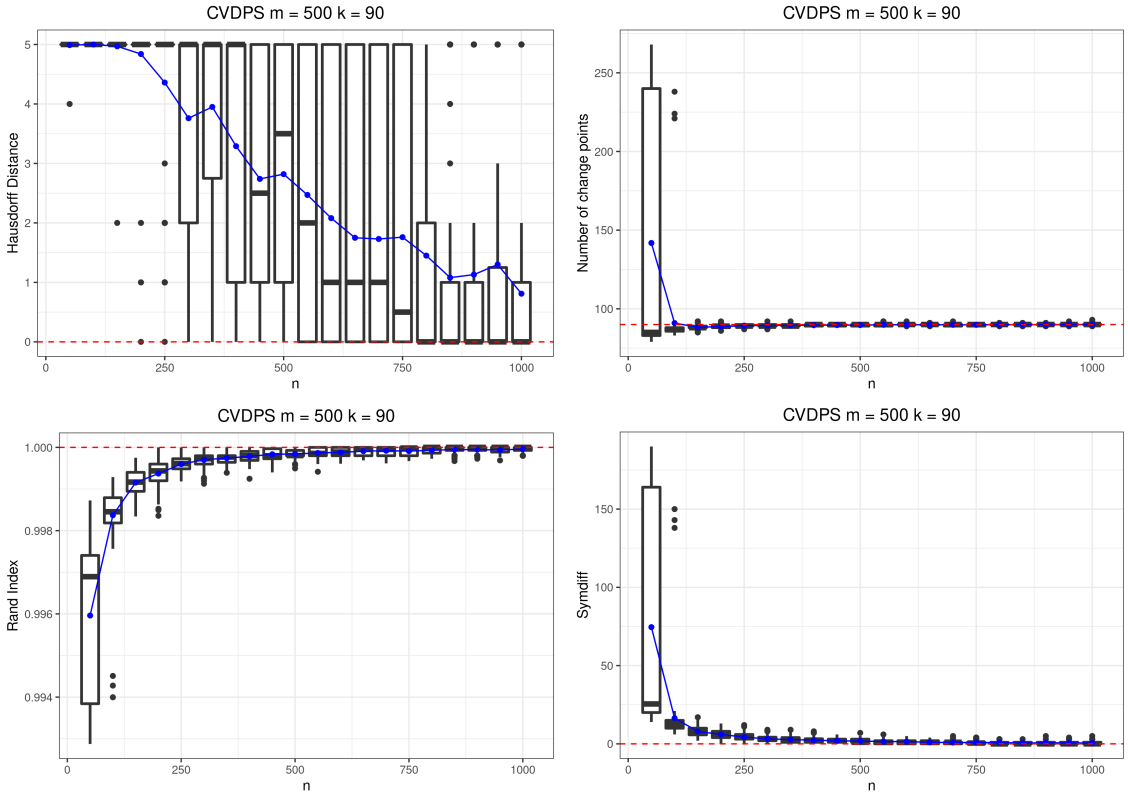


Figure E.42: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the CVDPS*

DPS LOG

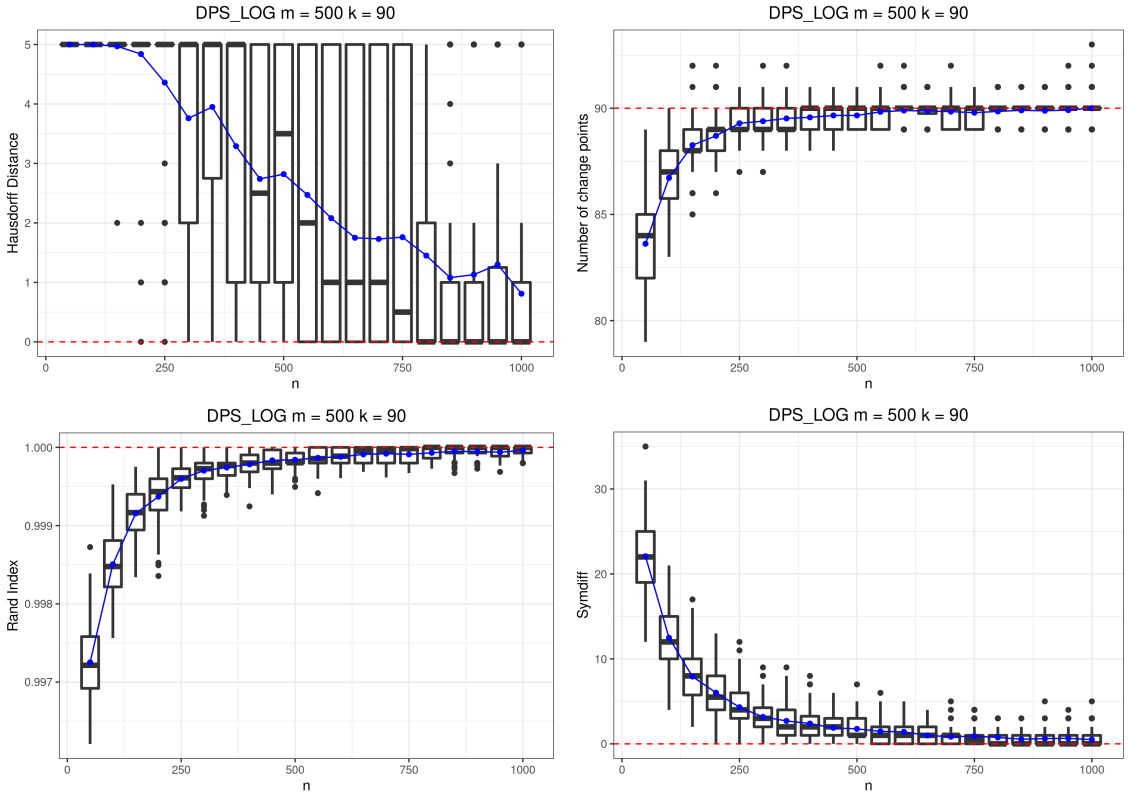


Figure E.43: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS LOG*

DPS SQRT

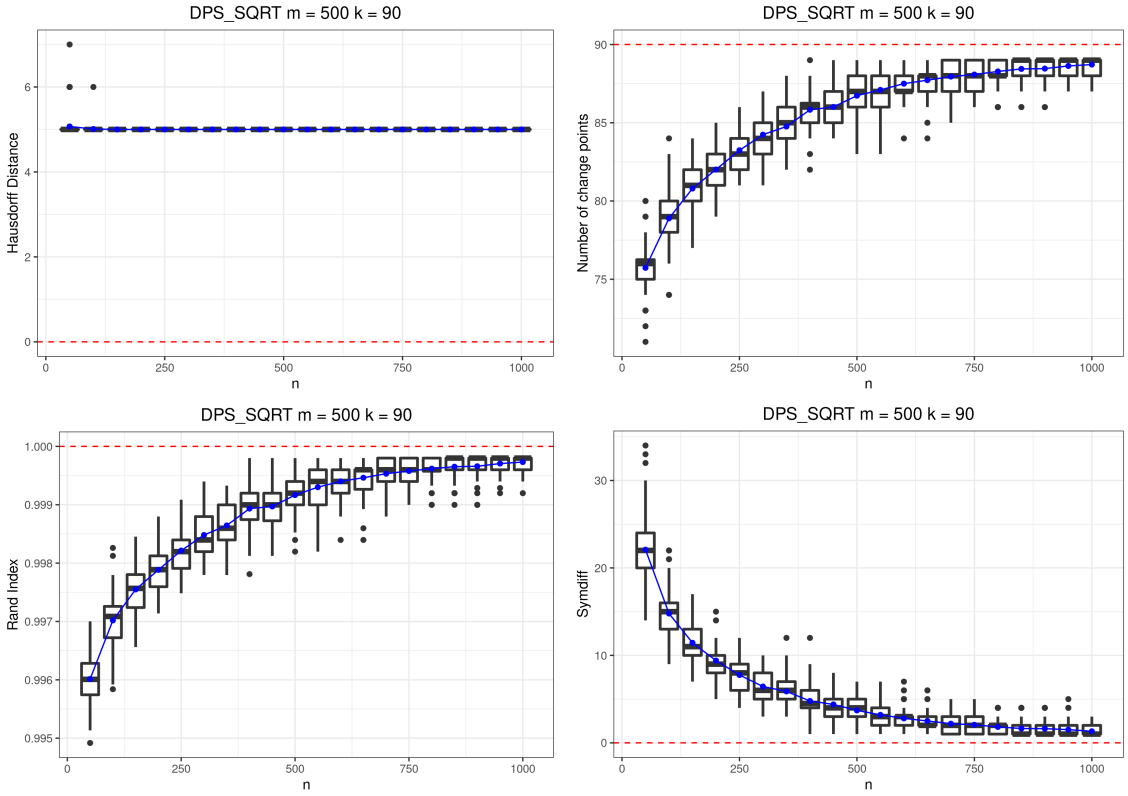


Figure E.44: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the DPS SQRT*

HS

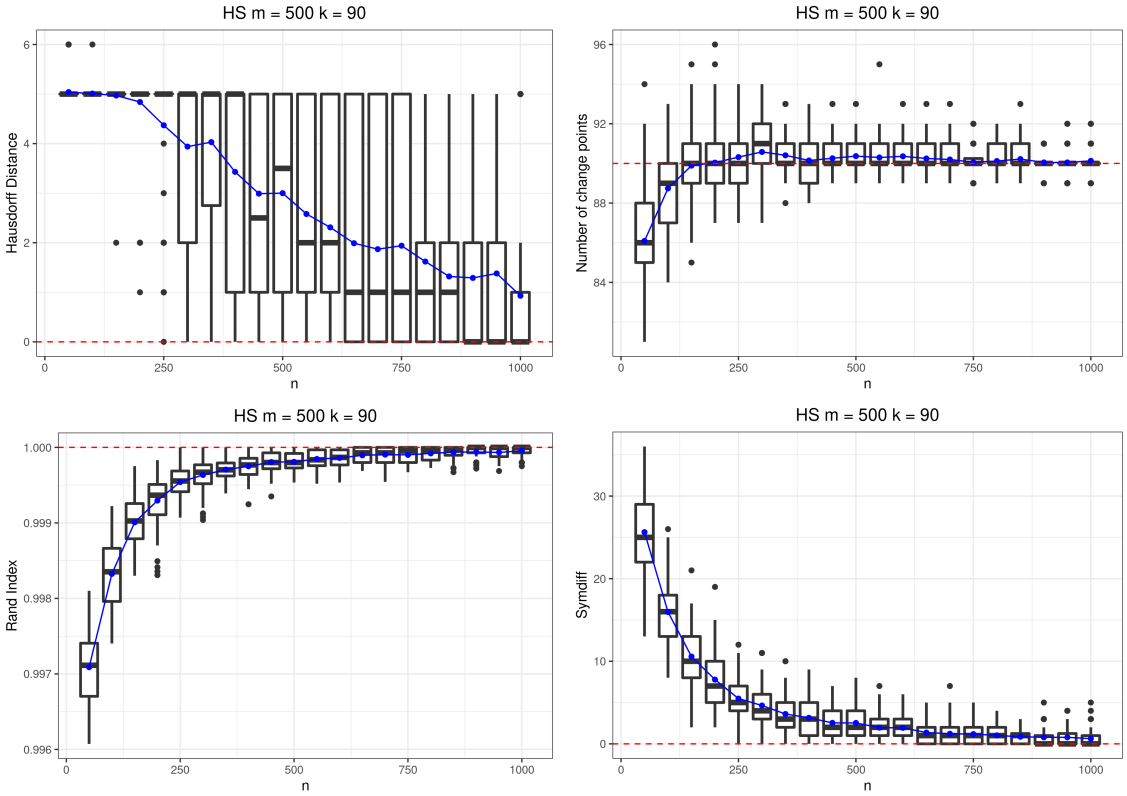


Figure E.45: *Number of change points estimated, Hausdorff distance, Rand Index and Symmetric difference for the HS*

Bibliography

Adams, R. P. and MacKay, D. J. C. [2007], 'Bayesian online changepoint detection'. 2

Agudelo-España, D., Gomez-Gonzalez, S., Bauer, S., Schölkopf, B. and Peters, J. [2020], 'Bayesian online prediction of change points'. 2, 3

Bouchikhi, I., Ferrari, A., Richard, C. and Bourrier, A. [2020], 'Online change-point detection with kernels'. 2

Carlin, B. P., Gelfand, A. E. and Smith, A. F. M. [1992], 'Hierarchical bayesian analysis of change-point problems', *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **41**(2), 389–405.
URL: *https://rss.onlinelibrary.wiley.com/doi/abs/10.2307/2347570* 2

Castro, B. M., Lemes, R. B., Cesar, J., Hünemeier, T. and Leonardi, F. [2018], 'A model selection approach for multiple sequence segmentation and dimensionality reduction', *Journal of Multivariate Analysis* **167**, 319–330. 3

Ceballos, F. C., Joshi, P. K., Clark, D. W., Ramsay, M. and Wilson, J. F. [2018], 'Runs of homozygosity: windows into population history and trait architecture', *Nature Reviews Genetics* **19**(4), 220. 25

Celisse, A., Marot, G., Pierre-Jean, M. and Rigaill, G. [2018], 'New efficient algorithms for multiple change-point detection with reproducing kernels', *Computational Statistics & Data Analysis* **128**, 200 – 220.
URL: *http://www.sciencedirect.com/science/article/pii/S0167947318301683* 3

Chen, J. and Gupta, A. K. [2011], *Parametric statistical change point analysis: with applications to genetics, medicine, and finance*, Springer Science & Business Media. 2

Cobb, G. W. [1978], 'The problem of the nile: Conditional solution to a changepoint problem', *Biometrika* **65**(2), 243–251. 3

Harchaoui, Z. and Cappe, O. [2007], Retrospective mutiple change-point estimation with kernels, *in* '2007 IEEE/SP 14th Workshop on Statistical Signal Processing', pp. 768–772. 2

Kim, S.-J., Koh, K., Boyd, S. and Gorinevsky, D. [2009], '\ell_1 trend filtering', *SIAM review* **51**(2), 339–360. 15

Kirin, M., McQuillan, R., Franklin, C. S., Campbell, H., McKeigue, P. M. and Wilson, J. F. [2010], 'Genomic runs of homozygosity record population history and consanguinity', *PloS one* **5**(11), e13996. 25, 28

Lee, T.-S. [2010], 'Change-point problems: Bibliography and review', *Journal of Statistical Theory and Practice* **4**(4), 643–662.
URL: *https://doi.org/10.1080/15598608.2010.10412010* 2

Lemes, R. B., Nunes, K., Carnavalli, J. E., Kimura, L., Mingroni-Netto, R. C., Meyer, D. and Otto, P. A. [2018], 'Inbreeding estimates in human populations: Applying new approaches to an admixed brazilian isolate', *PloS one* **13**(4), e0196360. 25

Leutenegger, A.-L., Sahbatou, M., Gazal, S., Cann, H. and Génin, E. [2011], 'Consanguinity around the world: what do the genomic data of the hgdp-ceph diversity panel tell us?', *European Journal of Human Genetics* **19**(5), 583–587. 25

Levy-leduc, C. and Harchaoui, Z. [2008], Catching change-points with lasso, *in* J. Platt, D. Koller, Y. Singer and S. Roweis, eds, 'Advances in Neural Information Processing Systems', Vol. 20, Curran Associates, Inc., pp. 617–624.
URL: *https://proceedings.neurips.cc/paper/2007/file/e5841df2166dd424a57127423d276bbe-Paper.pdf* 2

Li, G., Wang, J., Liang, J. and Yue, C. [2018], 'The application of a double cusum algorithm in industrial data stream anomaly detection', *Symmetry* **10**(7), 264. 3

Li, J. Z., Absher, D. M., Tang, H., Southwick, A. M., Casto, A. M., Ramachandran, S., Cann, H. M., Barsh, G. S., Feldman, M., Cavalli-Sforza, L. L. et al. [2008], 'Worldwide human relationships inferred from genome-wide patterns of variation', *science* **319**(5866), 1100–1104. 28

McQuillan, R., Leutenegger, A.-L., Abdel-Rahman, R., Franklin, C. S., Pericic, M., Barac-Lauc, L., Smolej-Narancic, N., Janicijevic, B., Polasek, O., Tenesa, A. et al. [2008], 'Runs of homozygosity in european populations', *The American Journal of Human Genetics* **83**(3), 359–372. 28

Niu, Y. S., Hao, N. and Zhang, H. [2016], 'Multiple change-point detection: A selective overview', *Statist. Sci.* **31**(4), 611–623.
URL: *https://doi.org/10.1214/16-STS587* 3

Page, E. S. [1954], 'Continuous inspection schemes', *Biometrika* **41**(1/2), 100–115.
URL: *http://www.jstor.org/stable/2333009* 2, 3

Page, E. S. [1955], 'A test for a change in a parameter occurring at an unknown point', *Biometrika* **42**(3-4), 523–527.
URL: *https://doi.org/10.1093/biomet/42.3-4.523* 2

Pemberton, T. J., Absher, D., Feldman, M. W., Myers, R. M., Rosenberg, N. A. and Li, J. Z. [2012], 'Genomic patterns of homozygosity in worldwide human populations', *The American Journal of Human Genetics* **91**(2), 275–292. 25

Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A., Bender, D., Maller, J., Sklar, P., De Bakker, P. I., Daly, M. J. et al. [2007], 'Plink: a tool set for whole-genome association and population-based linkage analyses', *The American journal of human genetics* **81**(3), 559–575. 28

Raftery, A. E. and Akman, V. E. [1986], 'Bayesian analysis of a poisson process with a change-point', *Biometrika* **73**(1), 85–89.
URL: *http://www.jstor.org/stable/2336274* 2

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.
URL: *https://doi.org/10.1175/JAM2493.1* 3

Shao, J. [1993], 'Linear model selection by cross-validation', *Journal of the American statistical Association* **88**(422), 486–494. 9

Tahmasbi, R. and Rezaei, S. [2008], 'Change point detection in garch models for voice activity detection', *IEEE Transactions on Audio, Speech, and Language Processing* **16**(5), 1038–1046. 3

Tartakovsky, A. G., Rozovskii, B. L., Blazek, R. B. and Hongjoong Kim [2006], 'A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods', *IEEE Transactions on Signal Processing* **54**(9), 3372–3382. 3

Tibshirani, R. [1996], 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society: Series B (Methodological)* **58**(1), 267–288. 10

Tibshirani, R. J., Taylor, J. et al. [2011], 'The solution path of the generalized lasso', *The Annals of Statistics* **39**(3), 1335–1371. 10

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. 10

Truong, C., Oudre, L. and Vayatis, N. [2020], 'Selective review of offline change point detection methods', *Signal Processing* **167**, 107299.
URL: *http://dx.doi.org/10.1016/j.sigpro.2019.107299* 3, 15, 21

van den Burg, G. J. J. and Williams, C. K. I. [2020], 'An evaluation of change point detection algorithms'. 3

Verbesselt, J., Hyndman, R., Newnham, G. and Culvenor, D. [2010], 'Detecting trend and seasonal changes in satellite image time series', *Remote Sensing of Environment* **114**(1), 106 – 115.
URL: *http://www.sciencedirect.com/science/article/pii/S003442570900265X* 3

Williams, S., Parry, B. and Schlup, M. [1992], 'Quality control: An application of the cusum', *British Medical Journal* **304**, 1359–1361. 3

Zou, C., Yin, G., Feng, L. and Wang, Z. [2014], 'Nonparametric maximum likelihood approach to multiple change-point problems', *Ann. Statist.* **42**(3), 970–1002.
URL: *https://doi.org/10.1214/14-AOS1210* 2