FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Jakub Lasota

Wild Binary Segmentation

Department of Probability and Mathematical Statistics

Supervisor of the bachelor thesis:	doc. RNDr. Michal Pešta, Ph.D.
Study programme:	Financial Mathematics
Study branch:	Financial Mathematics

Prague 2024

I would like to thank doc. RNDr. Michal Pešta, PhD. for many valuable tips and recommendations, for always being positive, and also for the time he dedicated to my work.

Název práce: Divoká Binární Segmentace

Autor: Jakub Lasota

Katedra: Katedra Pravděpodobnosti a Matematické Statistiky

Vedoucí bakalářské práce: doc. RNDr. Michal Pešta, Ph.D., Katedra Pravděpodobnosti a Matematické Statistiky

Abstrakt: Cílem této práce je popsat a vysvětlit několik metod pro detekci bodů změny, které se snaží odhadnout celkový počet a polohy strukturálních změn v datech. Z rozsáhlé škály dostupných metod se soutředíme pouze na konkrétní dvě, binární segmentaci a divokou binární segmentaci. Pro lepší pochopení obsahuje práce také několik ilustračních příkladů, které se snaží ukázat výhody a nevýhody každé z metod. Praktická část práce se soustředí na použití a porovnání obou metod v závislosti na volbě parametrů na denních logaritmických výnosech akcií společnosti Zoom Video Communications.

Klíčová slova: detekce bodů změny, více bodů změny, binární segmentace, randomizované algoritmy, informační kritérium

Title: Wild Binary Segmentation

Author: Jakub Lasota

Department: Department of Probability and Mathematical Statistics

Supervisor: doc. RNDr. Michal Pešta, Ph.D., Department of Probability and Mathematical Statistics

Abstract: The goal of this work is to describe some (multiple) change-point detection methods that aim to estimate the total number and locations of structural changes in the data. From the variety of all change-point detection methods, only binary segmentation and wild binary segmentation are explained. To enhance the understanding, the work contains a few illustrative examples that try to show the strengths and weaknesses of each method. The practical part of the work focuses on using and comparing both methods with various parameter choices on daily logarithmic returns of the Zoom Video Communications stock.

Keywords: change point-detection, multiple change-points, binary segmentation, randomized algorithms, information criterion

Contents

1. Introduction

Let $X_1, X_2, ..., X_T$ be observations taken over time T. Suppose we have a stochastic process $\{X_t, t \in \mathbb{Z}\}$ representable also as

$$X_t = f_t + \varepsilon_t, \; t \in \mathbb{Z},$$

where f_t is a deterministic one-dimensional piece-wise constant function and ε is a noise (also called error). It is of interest to find all locations of structural changes in this time series, as well as the total number of such changes. This approach is in literature usually called change-point detection, often as well as multiple change-point detection, where by change-point we mean the moment in time where a shift occurs.

Our goal is to estimate the number and locations of change-points in the data, under assumptions on the minimum permitted magnitude of the jumps in change-points, the distance of the points, or the total number N of these points, etc.

To achieve our goal we introduce two methods, (standard) binary segmentation and wild binary segmentation, both widely used for change-point detection problems.

Binary segmentation (BS) is a method, where we search the whole data set, at first, for only one change point. After this change-point is found, we split the interval (hence called segmentation) into two new intervals (hence called binary) and recursively repeat this on both newly created segments defined by the splitting point until the recursion is stopped. This method is simple to implement, even for a non-expert. One of the drawbacks of this method is that in each step it is looking only for one change point at a time. This results in binary segmentation being unsuitable for specific functions having unfavorable change-point configurations. Additionally, it can be shown that minimum spacings between adjacent change-points is of order greater than $T^{3/4}$ in the case of jump magnitudes being bounded away from zero.

Using the benefits of standard binary segmentation we later describe a new method called wild binary segmentation (WBS) introduced in Fryzlewicz [2014]. WBS tries to eliminate weaknesses of standard BS and further improve its performance.

In what follows, we at first motivate the problem and both procedures. Then we define both BS and WBS and describe their recursive behavior. Additionally, we illustrate BS and WBS on simple examples to enhance the understanding of both methods. At the end of the work, we show both method's performance on a real data example.

2. Motivation

Throughout this work we consider a time series model with observations X_t:

$$X_t = f_t + \varepsilon_t, \quad t = 1, ..., T, \tag{2.1}$$

where f_t is a deterministic, one-dimensional, piecewise constant function and ε is a variable describing some kind of a noise (error variable). We will denote the number of change-points by N and their locations by $\eta_1, ..., \eta_N$. Both N and locations $\eta_1, ..., \eta_N$ are unknown. Additional assumptions and conditions for f_t and ε_t we will specify later. For simplicity, we often call X_t 'data'.

2.1 The CUSUM Statistic

The key to both, standard and wild binary segmentation, is the CUSUM statistic defined by the inner product between the vector of observations $(X_s, ..., X_e)$ and particular vector:

$$\tilde{X}^b_{s,e} = \sqrt{\frac{e-b}{n(b-s+1)}} \sum_{t=s}^{b} X_t - \sqrt{\frac{b-s+1}{n(e-b)}} \sum_{t=b+1}^{e} X_t, \tag{2.2}$$

where $s \leq b < e$ and $n = e - s + 1$. We will call the vector $\tilde{X}^b_{s,e}$ the vector of 'contrast' weights. In this work, we consider this to hold for the whole time. For simplicity, we do not investigate the origin of this statistic nor its theoretical background any further.

2.2 Idea

The idea is simple. In the first step of binary segmentation, the algorithm computes $\tilde{X}^b_{1,T}$ and takes $b_{1,1} = arg \max_{b:1\leq b<T} \left|\tilde{X}^b_{1,T}\right|$ to be the first change-point candidate. If this candidate satisfies specific judging criterion, thus $b_{1,1}$ becomes our true change-point estimation, the whole interval $[1, T]$ is then split into two sub-intervals $[1, b_{1,1}]$ and $[b_{1,1} + 1, T]$. Because of such splits into two new intervals, we call this procedure binary segmentation. The algorithm then continues recursively by computing $\tilde{X}^b_{1,b_{1,1}}$ and $\tilde{X}^b_{b_{1,1}+1,T}$, possibly resulting in additional splits.

Such algorithms are often called 'top-down' algorithms (also divisive), splitting large intervals into smaller ones. There exist completely different approaches to this problem, for example, in Fryzlewicz [2018]. This procedure is often called 'bottom-up' (or also agglomerative) because it 'connects small intervals' that possibly have the same 'attributes'. In the sense of change-point detection, instead of splitting intervals in possible change-points, this method connects neighboring intervals that with high probability correspond to locally constant f_t.

Denote $\mathcal{F}^b_{s,e}$ to be a set of all vectors on $[s, e]$ that have only one change-point being exactly in b. Having this we get

$$arg \max_{b:s\leq b<e} \left|\tilde{X}^b_{s,e}\right| = arg \min_{b:s\leq b<e} \min_{\bar{f}^b_{s,e}\in\mathcal{F}^b_{s,e}} \|X^e_s - \bar{f}^b_{s,e}\|^2_2.$$

This means, that if the true function f_t has only one change-points on $[s, e]$ then $\hat{b_0} = arg\max_{b:s\leq b<e} \left|\tilde{X}^b_{s,e}\right|$ is a least squares estimator of b_0 which is the same as maximum likelihood estimator if ε_t are i.i.d and having normal distribution.

Altogether, if f_t has only one change-point on its entire domain $[1, T]$, then the estimation of this change-point is very likely to be correct. Unfortunately, if the true function has more than one change-point, the algorithm trying to find only change-point on such a function is trying so while having a wrong model. This may cause binary segmentation to fail when analyzing data with unfavorable change-point configuration.

It can be shown that 'narrowing' the searching interval around a change-point is very likely to overcome this instability. In practice, we, of course, don't know the locations of change-points, thus we cannot choose s and e this way (if we knew that, we wouldn't have to try to estimate change-point locations in the first place). We also cannot try every combination of s and e possible due to computational complexity.

Leading from the discussion above, our main goal will be using this 'narrowing' trick to locate change-points, but with some additional rules. We will not try to search every possible interval $[s, e]$ but we will randomly draw several intervals $[s, e]$ and find our $arg\max_{b:s\leq b<e} \left|\tilde{X}^b_{s,e}\right|$ on each such sub-interval separately. Having a relatively large number of such sub-intervals will give us a high probability that at least one of such intervals will be bounded by s and e the way we described in the previous paragraph. In other words, drawing this interval randomly will hopefully bound our true change-point 'close enough' when also being bounded 'far enough' from other change-points. One can be surprised about this randomness approach because the needed number of random draws is 'not that large'.

This idea leads us to the new detection method called wild binary segmentation.

3. Theory

In this chapter, we provide theoretical background for both methods, binary and wild binary segmentation. The text will contain illustrative examples to facilitate understanding and show a visual side of the problem. We first specify the theoretical model, motivate both algorithms and formulate needed assumptions for the consistency theorem to hold. In this work, we will not show proofs of any theorem. All proofs and important discussion can be found in Fryzlewicz [2014].

3.1 Model definition

In this work, we consider one specific model. We assume that:

- the random error (noise) sequence $\{\varepsilon_t\}_{t=1}^{T} \sim \mathcal{N}(0,1)$ being i.i.d.,
- the function $\{f_t\}_{t=1}^{T}$ is bounded, which means that we assume $|f_t| < \bar{f} < \infty \, for \, t \in [1,T]$,

where $\sim \mathcal{N}(\cdot,\cdot)$ means 'having normal (Gaussian) distribution' with specific parameters and by i.i.d. we mean independent identically distributed random variables.

(3.1)

The first assumption (normal distribution of the noise) is necessary to mention for clarity and for technical convenience. It is very reasonable to extend this problem to n.i.n.i.d (non-independent, not identically distributed) non-normal noise (and vice versa), but in what follows, we do not consider any of those other cases. We also assume that $\mathrm{Var}\varepsilon_t$ is known because in practice it is very often easy to be estimated relatively well.

For additional convenience, we also assume that $\eta_0 = 0$ and $\eta_{N+1} = T$ (this means that we suppose we have change-point in time 0 and also T, being so at the very start and very end of our time series).

3.2 Binary Segmentation

To fully understand the wild binary segmentation concept, we first discuss standard binary segmentation. Before we begin, we have to make some additional assumptions.

In this section of the text, we assume that the minimum spacings between change points are greater or equal than δ_T, where $\delta_T \leq CT^{\Theta}$. We want C to be greater than zero and $\Theta \leq 1$.

For magnitudes of the jumps in change points we also need to assume them to be greater or equal than $\underline{f}_T$ where $\underline{f}_T \geq CT^{-\omega}$ for $\omega \geq 0$. Mathematically speaking we want the $\min_{i=1,\dots,N} f_i' \geq \underline{f}_T$ where $f_i' = |f_{\eta_i} - f_{\eta_i - 1}|$.

Those assumptions are linked together by equation $\Theta - \frac{\omega}{2} > \frac{3}{4}$.

(3.2)

It is important to say that we do not expect the total number of change points N to be bounded. We allow N to be as large as the minimum spacings between change points δ_T allow. Formally we have $N = N(T)$ and $\eta_i = \eta_i(T)$ for $i \in \{0, 1, ..., N+1\}$. For simplicity, we will use the notation N and η_i instead of $N(T)$ and $\eta_i(T)$.

Both standard BS and WBS can be simply described by recursive algorithms. Standard binary segmentation iteration steps can be illustrated as follows:

- choose values for parameters s, e and ζ_T,
- on the interval $[s, e]$ find the first change point estimation, denote it by b_0, where $b_0 := arg\ \max_{b: s \le b \le e-1} \left| \tilde{X}^b_{s,e} \right|$,
- If $\left| \tilde{X}^{b_0}_{s,e} \right| > \zeta_T$ then add it to the set of estimated change points,
- recursively repeat 2 previous steps on segments $[s, b_0]$ and $[b_0 + 1, e]$.

For this algorithm, we have a few stopping criteria. One is quite obvious, that is when $s - e < 1$. The other one is when $\left| \tilde{X}^{b_0}_{s,e} \right| < \zeta_T$. We call ζ_T a threshold parameter.

The standard for the binary segmentation algorithm is that it takes $s = 1$ and $e = T$, this means that in the first step of the algorithm, we are searching the whole time series for one change point that maximizes our CUSUM statistic.

Theorem 1 (Consistency of BS). *Let X_T follow our model* (2.1) *and suppose our assumptions* (3.1) *and* (3.2) *hold. Let N and $\eta_1, ..., \eta_N$ denote, respectively, the number and locations of change-points. Let $\hat{N}$ denote the number and $\hat{\eta_1}, ..., \hat{\eta_N}$ the locations, sorted in increasing order, of the change points estimates obtained by the standard binary segmentation algorithm. Let the threshold parameter satisfy $\zeta_T = c_1 T^\theta$ where $\theta \in (1-\Theta, \Theta - 1/2 - \omega)$ if $\Theta \in (\frac{3}{4}, 1)$, or $\zeta_T \ge c_2 \log^p T$ $(p > 1/2)$ and $\zeta_T \le c_3 T^\theta$ $(\theta < 1/2 - \omega)$ if $\Theta = 1$, for any positive constants c_1, c_2, c_3. Then there exist positive constants C, C_1 such that $P(\mathcal{A}_T) \ge 1 - C_1 T^{-1}$, where*

$$\mathcal{A}_T = \{\hat{N} = N; \max_{i=1,...,N} |\hat{\eta_i} - \eta_i| \le C\epsilon_T\}$$

with $\epsilon_T = T^2 \delta_T^{-2} (\underline{f}_T)^{-2} \log T$.

Proof of this theorem and interesting discussion can be found in the appendix of Fryzlewicz [2014].

Examples

Example (1). In this example, our goal is to estimate locations, also the number, of change-points in the data. To remark, we still consider our model (2.1), that is, one dimensional, piecewise constant function f_t with added noise ε_t such that $\mathbb{E}(\varepsilon) = 0$ and its variance is equal to 1. A detailed description of the data can be found in the appendix (A). We try to visualize our data with a graph shown in Figure (3.1) and corresponding true f_t shown in Figure (3.2).

Our goal is to estimate, if possible, the locations of change-points in the data. We can surely say that the number of true change-points $N = 1$, where the

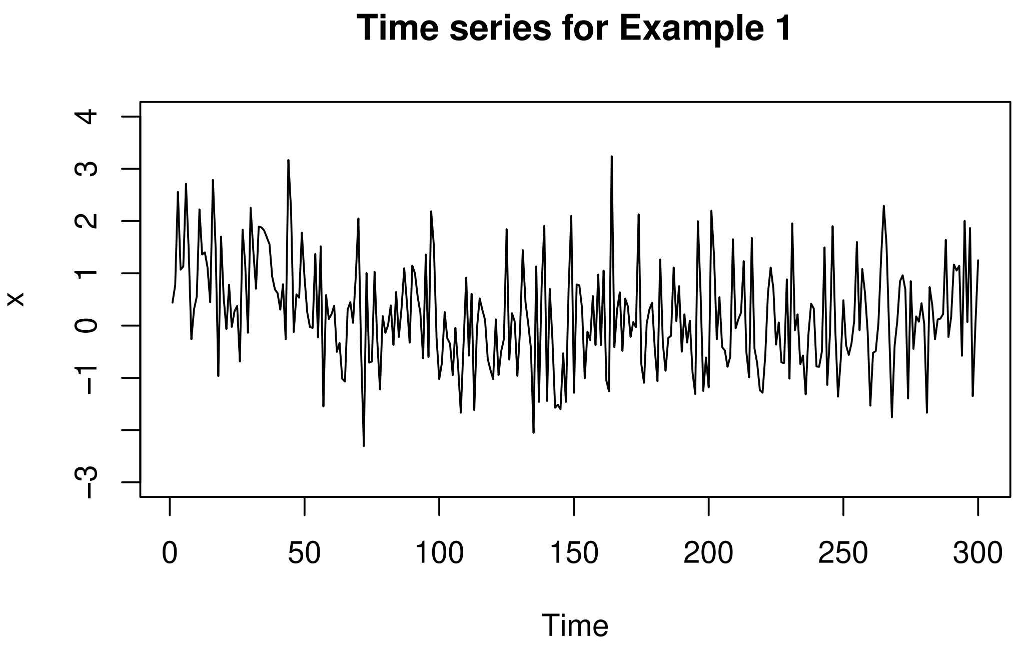

Figure 3.1: Time series for example 1

change-point occurs somewhere around $t = 50$. To verify this and to properly analyze the time series, we will use the R package 'wbs' created by Baranowski and Fryzlewicz [2019].

Following the BS algorithm, at first, the algorithm chooses s, e, starting and ending points for the searching interval. In the case of standard binary segmentation, we have $s = 1, e = 300$. On that interval we evaluate CUSUM statistic $\left|\tilde{X}^b_{s,e}\right|$ for each $b \in [s, e]$ while looking for b_0 such that $b_0 := arg\ \max_{b:s \leq b \leq e-1} \left|\tilde{X}^b_{s,e}\right|$.

After calling function $sbs()$ on our data, it returns us, for our specific example, a table with many rows containing various values. For comparison, the function $sbs()$ can also show us its estimation $\hat{f}_t$ of the true f_t shown in Figure (3.3).

We have here the first few rows of the table returned by $sbs()$. Again, a detailed description of obtaining these values is described in the corresponding section in appendix (A) of this work.

```
        s   e cpt        CUSUM      min.th scale
 [1,]   1   2   1 -0.233556067 0.233556067     6
 [2,]   1   7   2 -1.413027583 1.075023179     5
 [3,]   3   7   3  0.862911140 0.862911140     6
 [4,]   4   5   4 -0.041563273 0.041563273     8
 [5,]   4   7   5 -0.988092533 0.862911140     7
```

The first and second columns of our table describe starting and ending points s, e where the change-point candidate 'cpt' was found. The fourth column, which is called CUSUM, shows the maximal evaluated $\left|\tilde{X}^b_{s,e}\right|$ for $b \in [s, e]$ defined by the first two columns. The fifth column called 'min.th', shows us the minimal value of the threshold parameter ζ_T for which our change point would not be

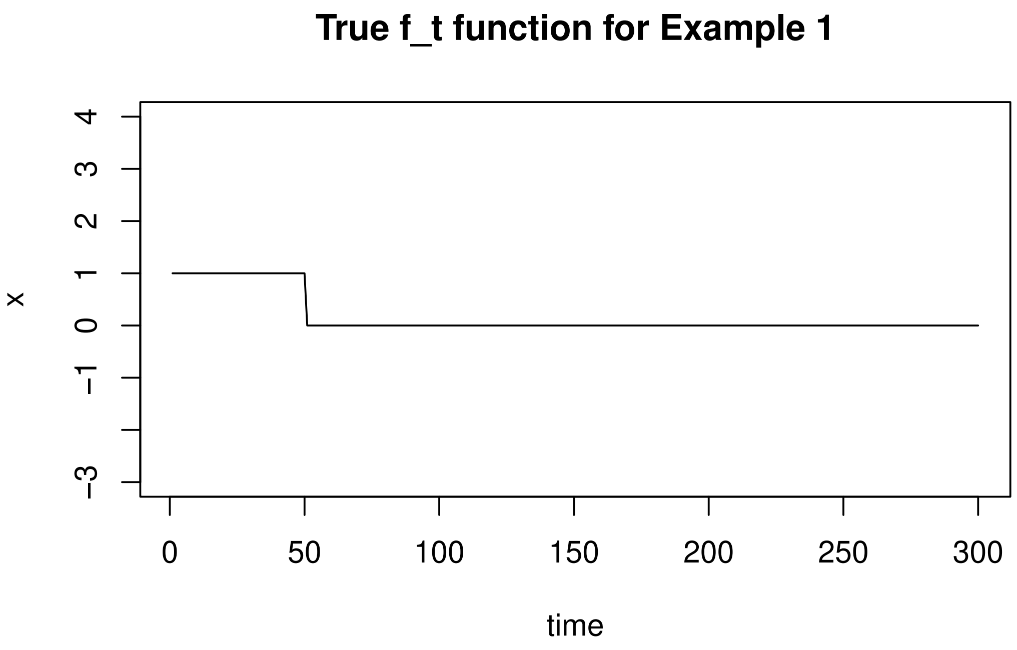

Figure 3.2: True f_t function for example 1

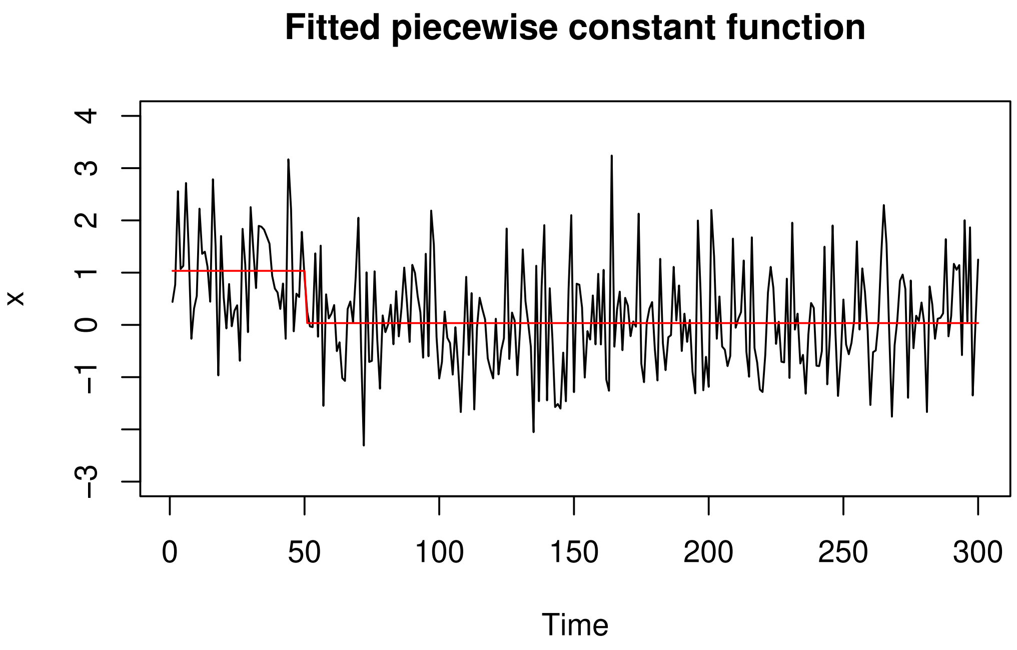

Figure 3.3: Time series for example 1 (black line) with $\hat{f}_t$ (red line)

considered significant, and thus ignored. The last column is not that simple to describe. In some sense, it tells us 'how many times have binary segmentation visited corresponding interval s, e'.

To find our 'best' (in terms of CUSUM statistic magnitude) change-point candidates more easily, we would like to have our CUSUM column sorted and evaluated in absolute value. Take a look at the new column called 'abs_cusum'. It is a transformation of our yet known column 'CUSUM' but transformed via R function *abs*() resulting in having CUSUM statistics in absolute values.

```
       s   e cpt        CUSUM      min.th scale   abs_cusum
[1,]   1 300  50  6.454678999 6.454678999     1 6.454678999
[2,] 164 173 164  2.997157068 0.942349367    10 2.997157068
[3,] 254 266 263 -2.496591469 1.043710975    11 2.496591469
[4,]  18  50  26 -2.467592950 1.189884295     3 2.467592950
[5,]  51  98  96 -2.428667577 1.374889043     4 2.428667577
```

Additionally, we have the table above sorted with respect to the new column 'abs_cusum' in decreasing order. As we can see, the maximal $\left|\tilde{X}_{s,e}^{b}\right|$ occurs at $t = 50$, while searching whole interval $[1, 300]$, exactly as we expected.

It is important to say that in this example we completely ignored ζ_T parameter choice and we also ignored the recursive behavior of the algorithm.

Example (2). In this example, we try to look at a slightly more complicated situation. Again, any details concerning example recreation will be shown in appendix (A). Our goal is to get change-point location estimates $\hat{\eta_1}, ..., \hat{\eta_N}$ and possibly $\hat{N}$ denoting the estimated number of change points. We also take a look at something new compared to the last example, that being threshold parameter choice and its influence.

We suppose that our model (2.1) still holds. Let us take a look at graph (3.4) below, which illustrates data for example 2.

The data consist of 300 values obtained in $t = 1, ..., T$, where $T = 300$. We can visually see some suspicious behavior approximately in the middle of the time interval. We use a standard binary segmentation algorithm using R implementation. Before we do so, allow us a little bit of cheating, and take a look at true f_t, that being, let us visualize the true change point locations $\eta_1, ...\eta_N$ and their number N in Figure (3.5).

In Figure (3.5) we can clearly see three change-points, one occurring somewhere around $t = 120$ other one being around $t = 140$ and the last one close to $t = 160$. In the appendix we expose that those numbers are precise, which means that our change-points are exactly at the locations mentioned before.

Following the standard binary segmentation algorithm, which we explained in section (3.2), we again, similarly to the previous example, choose s, e starting and ending points needed for algorithm initiation. The standard for classic binary segmentation is setting $s = 1$ and $e = T$ in this specific case, that being the whole time interval $[1, 300]$. We again compute CUSUM statistic $\left|\tilde{X}_{s,e}^{b}\right|$ for each $b \in [s, e]$ recursively continuing on possibly newly created intervals defined by previously found change-point.

On contrary to example 1 in (3.2) we will now also talk about threshold parameter ζ_t. The importance of this parameter was explained in the description

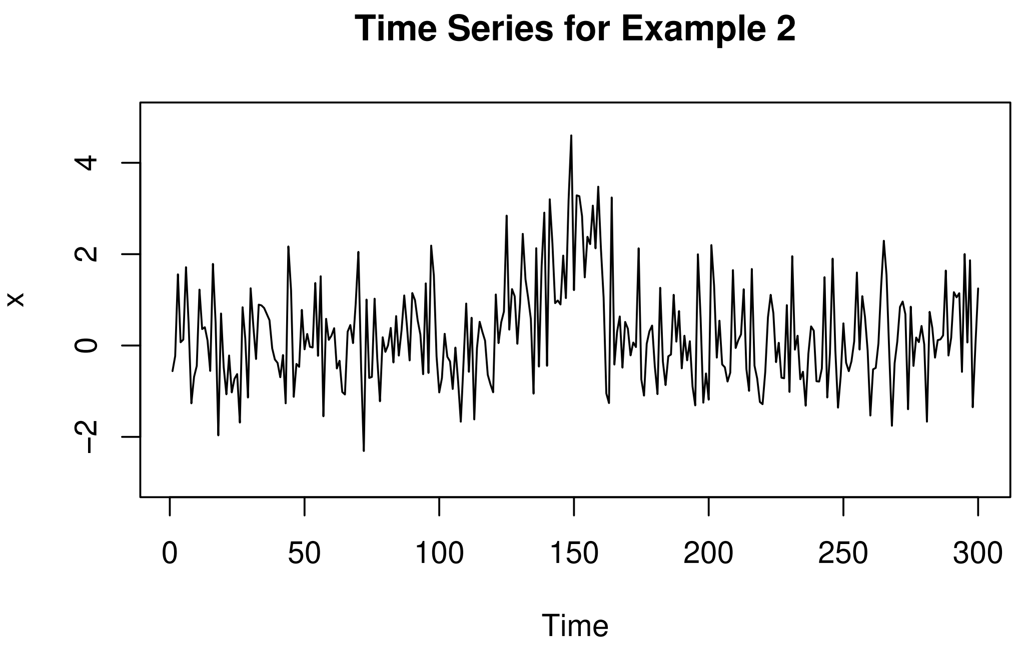

Figure 3.4: Time series for example 2

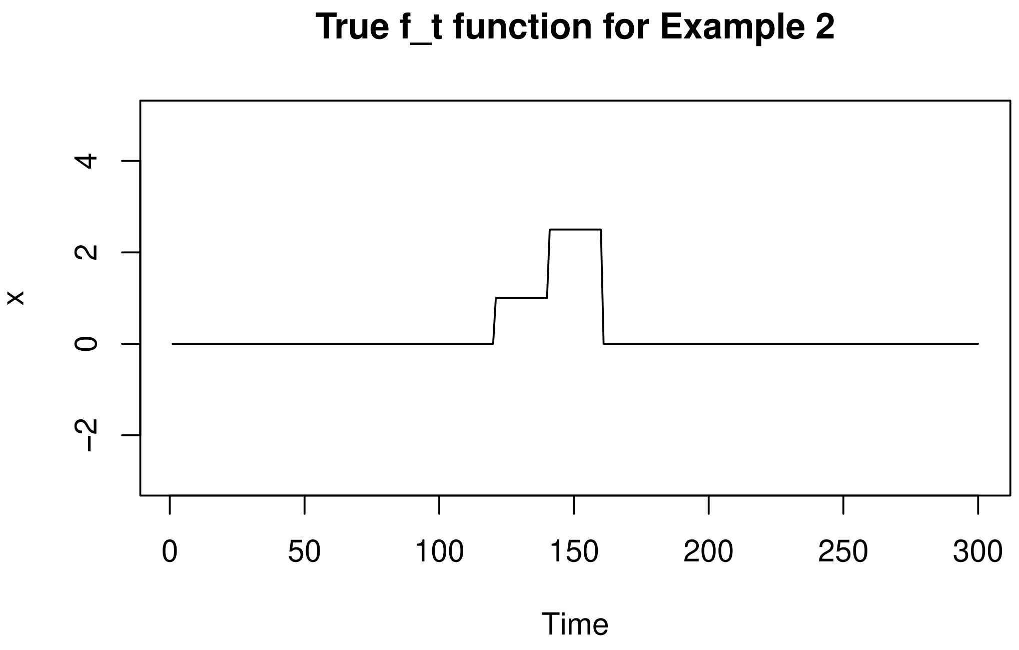

Figure 3.5: True f_t function for example 2

of the recursive idea of both, binary segmentation and wild binary segmentation, also its role illustration can be found in section (3.4). To sum up, the number of change-points estimations $\hat{N}$ and also possibly found change-point locations estimations $\hat{\eta_1}, ..., \hat{\eta_N}$ can be seen as a function of ζ_t. Thus, by choosing different values of parameter ζ_t we obtain different results. The discussion of parameter choice and its properties can be found in Fryzlewicz [2014], chapter 4 called 'parameter choice and simulation study'. From this section, we only extract the formula for parameter ζ_t and values for the previously mentioned constant C described in both sections about BS and WBS. For the remainder of the work, we will consider

$$\zeta_t = C\hat{\sigma}\sqrt{2}\log^{1/2} T,$$

where $\hat{\sigma}$ is median absolute deviation estimation of σ denoting the variance of standard normal noise. For this, we will use an R function called *mad*() which computes respective estimation $\hat{\sigma}$ needed for the evaluation of ζ_t, although it is expected to be close to 1 due to artificial data construction shown in (A). In the same part of the Fryzlewicz [2014], we can find the value for constant C, which we will use in this work as well, that is, from now on we consider $C = 1.3$.

Now we have everything needed to proceed. We are able to use the function *sbs*() on our data. We can take a look at the same table shown in example 1 in (3.2) but with values from example 2 (3.2).

```
         s   e cpt        CUSUM      min.th scale      abs_cusum
  [1,] 121 300 161  8.698185335 3.568530761     2    8.698185335
  [2,] 121 161 147 -4.449782798 3.568530761     3    4.449782798
  [3,]   1 300 120 -3.568530761 3.568530761     1    3.568530761
  [4,] 164 287 164  3.207762587 1.684802218     5    3.207762587
  [5,] 264 287 266  2.801439505 0.985121451     7    2.801439505
```

From this table we can just by looking at the column called 'abs_cusum' imply that we have very likely change-points candidates at $t = 161$, possibly at $t = 147$, maybe also at $t = 120$ and at $t = 164$. Let us now compare our assumptions with a graph that shows $\hat{f}_t$, that being an estimation of true f_t illustrated in Figure (3.6) made by R function *sbs*().

This figure shows no change-point estimations at all. Why did that happen? Now the discussion about threshold parameter ζ_t comes in handy. Let us take a look at the 'min.th' column shown in the table above. If we sort the table by 'min.th' in decreasing order, we get the same first 5 rows as shown above. That being, the highest value is approximately 3.57. How did we get this number and how does it influence our result? Before we answer this question, let us take a closer look at ζ_t.

As we mentioned previously, we consider $\zeta_t = C\hat{\sigma}\sqrt{2}\log^{1/2} T$. Proceed in order, we have $C = 1.3$, R function *mad*() returned us $\hat{\sigma} = 1.006067$ approximately. Now for ζ_t we get $\zeta_t = 1.3 * 1.006067 * \sqrt{2} * \log^{1/2}(300) \doteq 4.417399$.

It might seem strange. Why do we have no change-point estimations, when we clearly see at least one and maybe even two change points, which passed the threshold ζ_t speaking of their CUSUM magnitude $\left|\tilde{X}^b_{s,e}\right|$. The problem here is the recursive behavior of the BS procedure. In the first step of algorithm we search the whole time interval, looking for change-points. In this case, we have

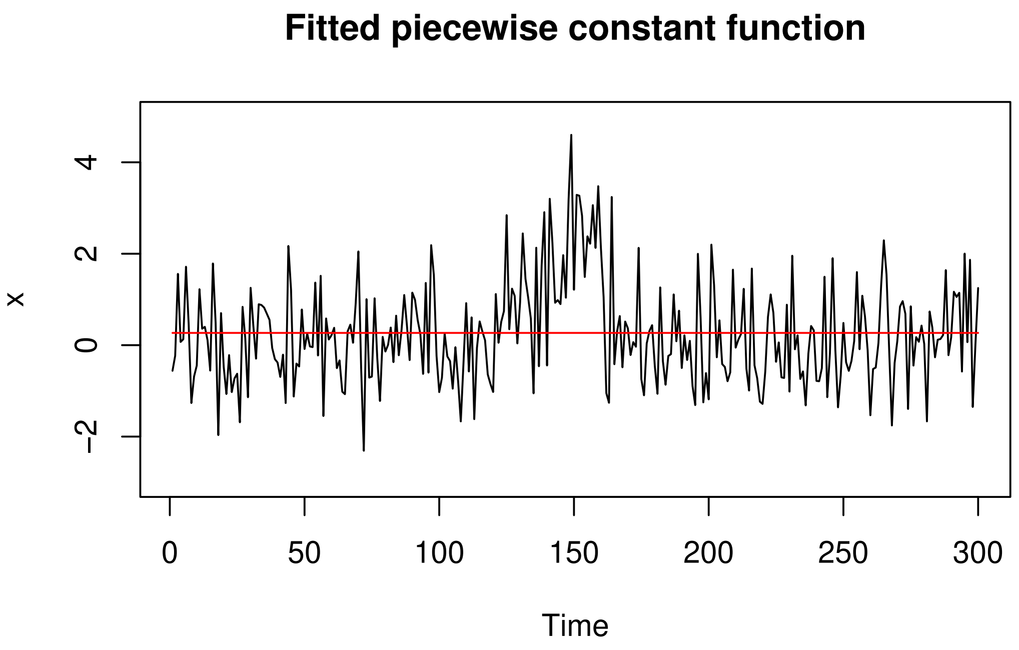

Figure 3.6: Time series for example 2 (black line) with $\hat{f}_t$ (red line)

```
          s   e cpt        CUSUM      min.th scale      abs_cusum
   [1,]   1 300 120 -3.568530761 3.568530761     1    3.568530761
```

describing our first iteration of *sbs*() function algorithm. The algorithm, described in section (3.2), compares the maximal $\left|\tilde{X}^b_{s,e}\right|$ found on interval s, e with ζ_t. In this case, it checks if $\left|\tilde{X}^{120}_{1,300}\right| > \zeta_t$. Specifically, it verifies whether approximately $3.568 > 4.417$. This clearly is not satisfied, even though we rounded both numbers. That means, that the algorithm stops in the first iteration resulting in finding no change points due to ζ_t being too high even for the first step change-point estimate.

The reason of the failure is the true change points were simply 'too difficult' for standard binary segmentation to find. We could either make the interval $[s, e]$ more narrow or change the threshold parameter ζ_t to overcome this unfavorable situation. We will skip the narrowing variant for the sake of wild binary segmentation illustration.

If we lower the threshold parameter to $\zeta'_t < 3.568530761$, for example, $\zeta'_t = 3.5$ we get a completely different result. After choosing $\zeta_t = \zeta'_t$ we would get all the first three points from the table as change-points estimations, that is, $\hat{N} = 3$, $\hat{\eta}_1 = 120, \hat{\eta}_2 = 147$ and $\hat{\eta}_3 = 161$.

The influence of parameter ζ_t choice is essential to analyze the data correctly and to get the needed result. It is important to choose this parameter cautiously. In the last part of the example, where we introduced ζ'_t, we used the knowledge of the threshold value needed to get the result that we want. This approach can be considered unfair because we should not choose our parameters after the analysis is done but beforehand.

3.3 Wild Binary Segmentation

This section tells us more about wild binary segmentation. We call it wild because, in what follows, we need a random draw of sub-intervals from $[1, T]$. First, we denote F_M^T to be the set of random intervals. Those random intervals we denote by $[s_m, e_m]$ where $m = 1, ..., M$ whose, as we mentioned above, start and end points are drawn independently and uniformly from the set $1, ..., T$. What is and how to choose parameter M we discuss later. Similarly, as we shown the BS algorithm, we can illustrate the WBS algorithm as well:

- Select starting values for parameters s, e and ζ_T,
  (same as BS)
- create set $\mathcal{M}_{s,e}$ of those indices m for which $[s_m, e_m] \in F_T^M$ is such that $[s_m, e_m] \subseteq [s, e]$
  (new to WBS)
- (optional) expand $\mathcal{M}_{s,e} := \mathcal{M}_{s,e} \cup \{0\}$, where $[s_0, e_0] = [s, e]$
  (new to WBS)
- define $(m_0, b_0) := arg\max_{m \in \mathcal{M}_{s,e}, b \in \{s_m, ..., e_m - 1\}} \left| \tilde{X}^b_{s_m, e_m} \right|$,
  (similar to BS)
- if $\left| \tilde{X}^{b_0}_{s_{m_0}, e_{m_0}} \right| > \zeta_T$ then add it to the set of estimated change points,
  (same as BS)
- recursively repeat previous steps on segments $[s, b_0]$ and $[b_0 + 1, e]$.
  (same as BS)

The optional step is there to make sure that our procedure also searches the entire interval $[s, e]$ and not only its randomly drawn sub-intervals. Especially when $[s, e]$ has only one change point, it is on point to examine this whole interval.

We can see a few stopping criteria for our algorithm. One is again being $s - e < 1$. The other occurs when the $\left| \tilde{X}^{b_0}_{s_{m_0}, e_{m_0}} \right| < \zeta_T$. We call ζ_T a threshold parameter.

It is also interesting to say that, unlike BS, this procedure without the optional step returns us estimated change points in decreasing order with respect to maxima of $\left| \tilde{X}^{b_0}_{s_{m_0}, e_{m_0}} \right|$. This is due to maximization over m. Maxima of CUSUM test statistics are not obliged in any way to come sorted for standard BS.

Now we might wonder, why do we use randomly drawn intervals instead of fixed ones. In some cases change points require 'narrow' intervals $[s, e]$ to be detectable. When we randomly draw sub-intervals on $[s, e]$, we have always a probability greater than zero that we will draw narrow enough intervals around those 'difficult to find' change points in the set F_M^T.

On the other hand, let us consider fixed intervals. Let us also say that start and end points $[s_m, e_m]$ can take all possible values from $1, ..., T$. Additionally, to be fair, say that the number of intervals will be the same as in the random scenario. Then we can expect at least some of the fixed intervals to be longer enough than the corresponding random intervals that they will not be able to detect those 'difficult to find' change points.

Another advantage of the random approach for interval selection is that when we find out that the number of such drawn intervals is insufficient. In the fixed design, we should consider redrawing the whole set of sub-intervals. But in the case of random choice, the only thing we need to do is simply draw an additional number of intervals from the previously chosen distribution.

When we make sure that our number of sub-intervals M is large enough, the difference between fixed and random approach can be expected to be minimal. This idea of drawing random intervals from s, e to overcome unfavorable change-point configuration was taken further in Baranowski et al. [2019]. Here, instead of choosing $(m_0, b_0) := arg \max_{m \in \mathcal{M}_{s,e}, b \in \{s_m, ..., e_m - 1\}} \left| \tilde{X}^b_{s_m, e_m} \right|$ as the 'best' change point candidate, this method chooses that (m_0, b_0) as change point candidate, whose 'contrast' exceeds specific threshold and has the smallest $|s_m - e_m|$, where $[s_m, e_m] \subseteq [s, e]$ in the sense we described above. The idea is that the smaller interval s_m, e_m is the higher the chance that it will contain only one 'feature'. In our case, that is only one change-point.

Before we proceed further, similarly to standard BS, we need to formulate some assumptions for WBS to be consistent.

Spacings between change point satisfies $\min_{i=1,...,N+1} |\eta_i - \eta_{i-1}| \geq \delta_T$. For the magnitudes we assume $f'_i = |f_{\eta_i} - f_{\eta_{i-1}}|$ to satisfy $\min_{i=1,...,N} f'_i \geq \underline{f}_T$, where δ_T and $\underline{f}_T$ are linked by the equation $\delta_T^{1/2} \underline{f}_T \geq C \log^{1/2} T$ for a constant C large enough.

(3.3)

Theorem 2 (Consistency of WBS). *Let X_T follow model* (2.1) *and suppose our assumptions* (3.1) *and* (3.3) *hold. Let N and $\eta_1, ..., \eta_N$ denote, respectively, the number and locations of change-points. Let $\hat{N}$ denote the number and $\hat{\eta}_1, ..., \hat{\eta}_N$ the locations, sorted in increasing order, of the change point estimates obtained by the wild binary segmentation algorithm. There exist two constants $\underline{C}, \overline{C}$ such that if $\underline{C} \log^{1/2} T \leq \overline{C} \delta_T^{1/2}$, then $P(\mathcal{A}_T) \geq 1 - C_1 T^{-1} - T\delta_T^{-1}(1 - \delta_T^2 T^{-2}/9)^M$, where*

$$\mathcal{A}_T = \{\hat{N} = N; \max_{i=1,...,N} |\hat{\eta}_i - \eta_i| \leq C \log T (\underline{f}_T)^{-2}\}$$

for certain positive constants C, C_1.

Proof of this theorem can be found in the appendix of Fryzlewicz [2014].

Now we discuss some properties of M being the (minimal) number of random draws needed to the speed of convergence $P(\mathcal{A})$ to 1 be suitably bounded. We naturally expect the number M to rise with decreasing δ_T. Mathematically we would like to

$$T\delta_T^{-1}(1 - \delta_T^2 T^{-2}/9)^M \leq T^{-1}$$

in order it to be the same rate as $C_1 T^{-1}$. After few steps we get

$$M \geq \frac{9T^2}{\delta_T^2 \log T^2 \delta_T^{-1}},$$

where we used the fact that $\log(1 - y) \approx y$ for y being relatively close to 0.

Another question we might ask ourselves is why we even use the recursive algorithm for WBS's random draws. Why don't we just take all change points estimations sorted by CUSUM statistic magnitude that passed threshold parameter ζ_t? In other words, why don't we just take all those points found by maximizing $\left|\tilde{X}^b_{s_m,e_m}\right|$ over all intervals $[s_e, s_m] \in F^M_T$. The answer is not that difficult. Doing this we might get some true change points estimated more times and in more positions. We avoid this problem by restricting us to one interval $[s_e, s_m]$ at a time.

Examples

Similarly to standard BS, we now take a look at some examples that hopefully help us better understand the new wild binary segmentation algorithm.

Example (3). In this example, we take a look at the case shown in Example 2, where BS 'failed'. To remark, we have data illustrated in (3.4) and corresponding true function f_t shown in (3.5). Following the algorithm described in section (3.3) we will use an R function called *wbs*() from the R package Baranowski and Fryzlewicz [2019]. The difference we see here is the 'narrowing' approach. As mentioned in section (3.3) we draw M random intervals on $[s, e]$. In chapter 4 of Fryzlewicz [2014] the author emphasizes the importance and influence of M being the number of (random) intervals used. He says that in the example with $T = 2000$ he used $M = 5000$ resulting in low computational time. Note, that the higher the number M the lower the influence of each random draw on the result. Choosing $M = 0$ results in a standard binary segmentation algorithm not benefiting from the random interval approach at all.

In what follows, we will consider $M = 5000$ even though it is a little bit of 'overkill' for our example.

After calling *wbs*() on the data, we get a similar table to examples 1 and 2.

```
       s   e cpt       CUSUM     min.th scale  abs_cusum
 [1,] 138 286 160  9.57997242 9.57997242     1 9.57997242
 [2,]   8 159 137 -9.12661371 9.12661371     2 9.12661371
 [3,]  99 132 120 -4.04060025 4.04060025     3 4.04060025
 [4,] 164 261 164  3.24375283 3.24375283     2 3.24375283
 [5,] 214 266 263 -3.07509067 3.07509067     3 3.07509067
```

Important to say, that function *wbs*() has many useful information we can look at, one being calculated threshold parameter ζ_t value for our specific case, as shown in Appendix (A). After extracting this information from the *wbs*() function, we obtain $\zeta_t \doteq 4.0123$.

In the first step, algorithm takes $s = 1, e = 300$ and draw random intervals on $[1, 300]$ calculating $\left|\tilde{X}^b_{s_m,e_m}\right|$ for each b and m possible. This results in $\left|\tilde{X}^{160}_{138,286}\right|$ being highest of them all, passing ζ_t with ease. Mathematically, we are checking if approximately $\left|\tilde{X}^{160}_{138,286}\right| = 9.58$ is greater than ζ_t. $[1, 300]$ is thus split into 2 new intervals defined by previously found change-point estimate at $t = 160$. The second and third iterations correspond with the second and fourth lines of our table ('scale' = 2). Now looking at $[1, 160]$ we randomly draw an interval

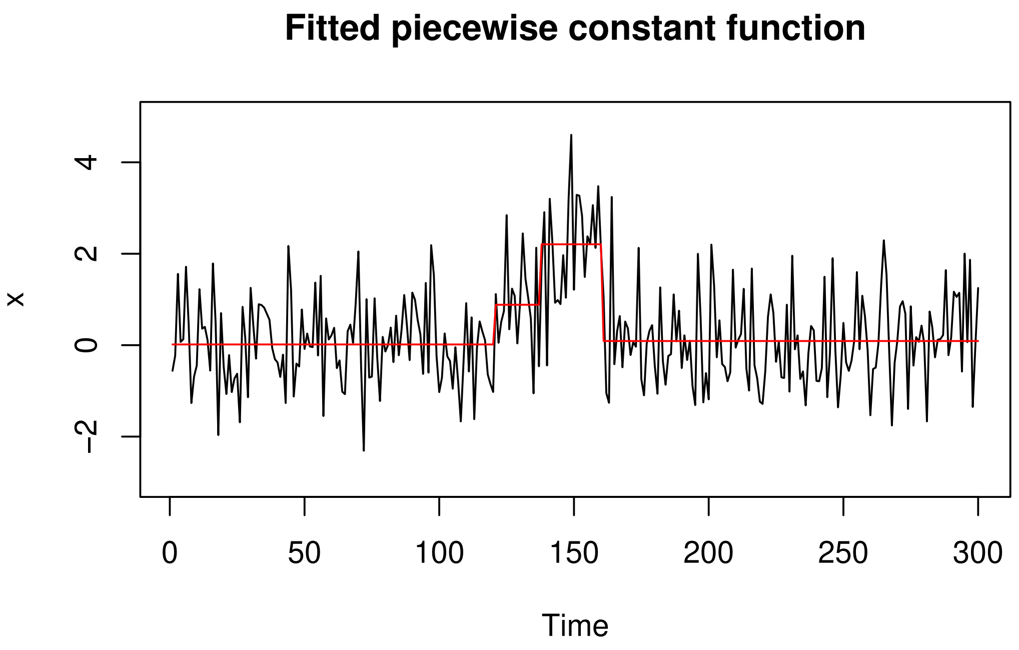

Figure 3.7: Time series for example 3 (black line) with $\hat{f}_t$ (red line) using WBS with $M = 5000$

$[8, 159]$ which again results in maximizing $\left|\tilde{X}^b_{s_m,e_m}\right|$ for randomly drawn $[s_m, e_m]$ on $[1, 160]$, that being the maximum occurs for $\left|\tilde{X}^{137}_{8,159}\right|$ again passing threshold check without any trouble (checking if approximately $\left|\tilde{X}^{137}_{8,159}\right| = 9.13$ is greater than ζ_t). For the second part of the second integration of the WBS algorithm, we now search $[161, 300]$ for any possible change-points. Evaluating $\left|\tilde{X}^b_{s_m,e_m}\right|$ for every b and m on randomly drawn sub-intervals from $[161, 300]$ results in maximum at $t = 164$. This unfortunately results in stopping iteration on this part of $[1, 300]$ since even the maximum of all CUSUMs on this part of the time series resulted only in $\left|\tilde{X}^{164}_{164,261}\right| \doteq 3.24$ being 'not good enough' for threshold parameter ζ_t. On the other hand, the previously found change point at $t = 137$ allows us to continue, resulting in another change-point estimation at $t = 120$ still, but very closely, passing the threshold check. Since all other CUSUMs are lower than ζ_t we stop the algorithm here, having true f_t estimation $\hat{f}_t$ shown in Figure (3.7) being very close to true f_t shown in Figure (3.5).

Example (4). Let us take a look at a little bit more complicated case. We still want to estimate locations and the number of change-points in the data. To achieve this, we will use the WBS algorithm with $M = 5000$ using the same argumentation as in Example 3. Data are visualized in Figure (3.8) and corresponding true f_t is shown in Figure (3.9). Detailed construction of the data and whole example is shown in Appendix (A). This example considers a very unfavorable change-point configuration which we try to overcome with WBS.

Following the steps used in Example 3, we use R function *wbs*() taken from the R package called 'wbs' made by Baranowski and Fryzlewicz [2019]. R console printed the following table:

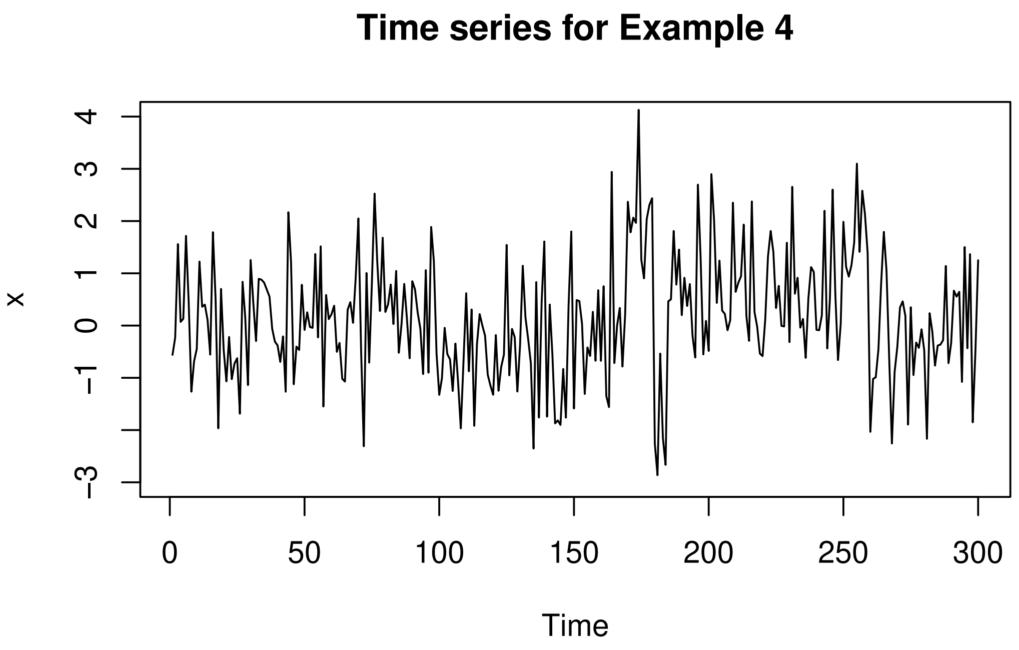

Figure 3.8: Time series for example 4

```
       s   e cpt        CUSUM      min.th scale   abs_cusum
[1,]   1 300 163 -7.778865287 7.778865287     1 7.778865287
[2,] 164 300 179  7.048670401 7.048670401     2 7.048670401
[3,] 180 300 184 -6.122027455 6.122027455     3 6.122027455
[4,] 185 300 259  5.793638369 5.793638369     4 5.793638369
[5,]   1 163  98  4.619866795 4.619866795     2 4.619866795
[6,] 164 179 169 -4.048986994 4.048986994     3 4.048986994
[7,] 185 259 249 -3.346755352 3.346755352     5 3.346755352
```

having still the same structure as described in previous examples. Extracting ζ_t parameter value from $wbs()$ function results in ζ_t being approximately equal to 4.2.

Extending the 'top-down binary-tree' idea explained in example 3 we can say that in step 1 ('scale' = 1) the algorithm finds a change-point at $t = 163$, passing the threshold, thus taken as valid. We split the interval into two new intervals defined by previously found change-point. Both these change points ('scale' = 2) again passed threshold parameter magnitude in terms of corresponding CUSUM statistic magnitude, thus also considered valid. In the third step, only one of four sub-intervals contains an acceptable change-point candidate, that being candidate at $t = 184$ found while searching interval $[s_m, e_m] = [180, 300]$. This allows the algorithm to find the last change-point available in terms of the thresholding approach at $t = 259$ successfully passing the threshold check. Since there are no other change points exceeding the threshold parameter value in terms of their CUSUM statistic magnitude, the algorithm ends here resulting in finding exactly 5 change-points. The result of WBS trying to estimate true f_t is shown in Figure (3.10).

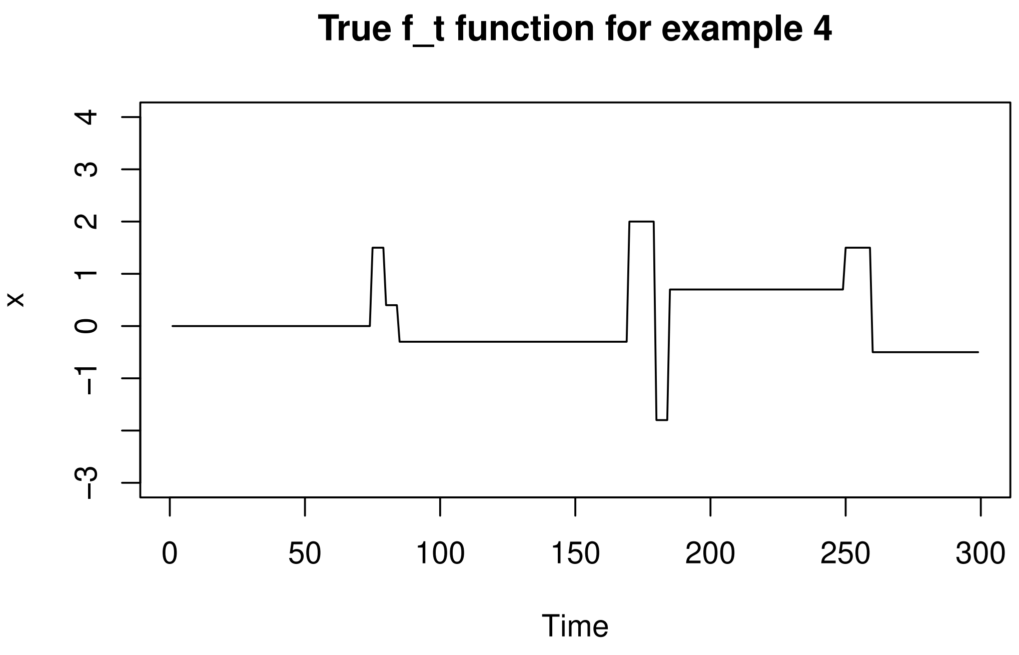

Figure 3.9: True f_t function for example 4

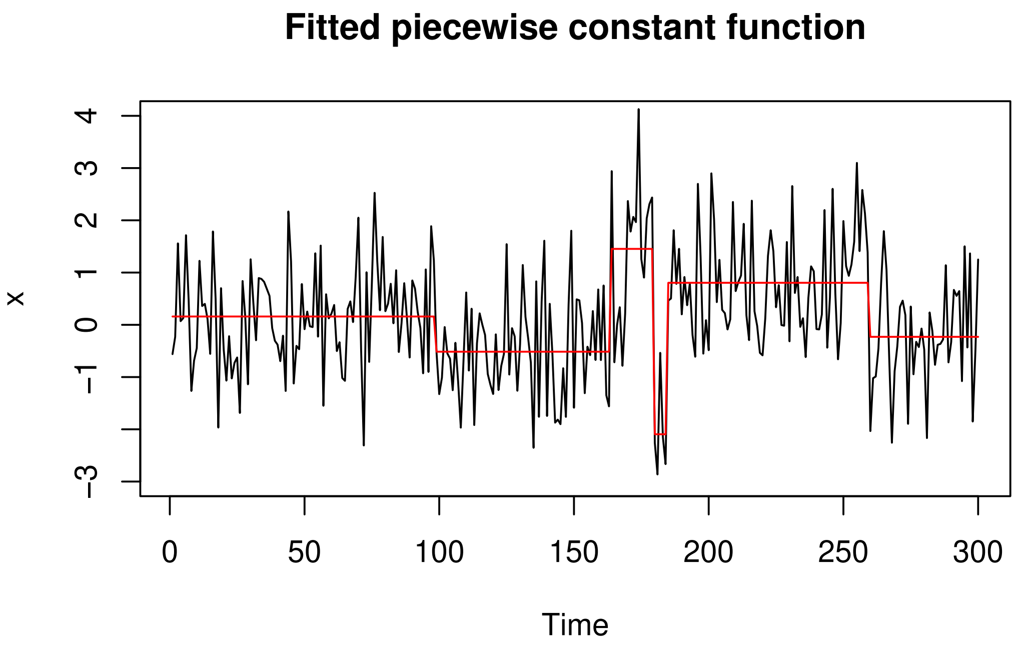

Figure 3.10: Time series for example 4 (black line) with $\hat{f}_t$ (red line) using WBS with M = 5000, thresholding approach with $\zeta_t \doteq 4.2$

This example again shows that the result we get is a function of parameter ζ_t. If we lower its value to be even a little bit less than 4, we would get different (maybe better) $\hat{f}_t$. In some cases the resulting $\hat{f}_t$ might be too sensible to very little changes in ζ_t magnitude, putting a person into a very difficult situation of 'correct' parameter value choice. It is important to say that as mentioned in previous examples, the parameter choice should be done before the data analysis, not vice-versa.

3.4 Strengthened Schwarz Information

In this section, we describe properties of the threshold parameter ζ_t. The text above implies that the amount of estimated change points is dependent on our selection of threshold parameter, that means, just for this section, we can denote $\hat{N} = \hat{N}(\zeta_t)$ and $\mathcal{C}(\zeta_t) = \{\hat{\eta}_1, ..., \hat{\eta}_{\hat{N}(\zeta_T)}\}$. $\hat{N}$ being a function of ζ_T, we can see that $\hat{N}$ is a non decreasing function of ζ_T. For the 'collection' set $\mathcal{C}$ holds that if $\zeta_T^1 < \zeta_T^2$ then $\mathcal{C}(\zeta_T^1) \subseteq \mathcal{C}(\zeta_T^2)$ almost surely. Consider a decreasing sequence $\{\zeta_T^k\}_{k=0}^K$ of thresholds such that $\left|\mathcal{C}(\zeta_T^k)\right| = k$ for a certain fixed constant K and also assume $N \leq K$. From this discussion, we can now choose the model in different ways. The first one is the obvious, selection of ζ_T itself, thus finding appropriate model $\mathcal{C}(\zeta_T)$. The new approach is choosing a sequence of candidates $\{\mathcal{C}(\zeta_T^k)\}_{k=0}^K$ and choosing one that satisfies certain conditions. Using the second method makes us completely free from choosing the threshold parameter ourselves. Also instead of evaluating the parameter on its own, now we can see it as a function of the number of change points candidates k. This being said we, only for this section, define $\mathcal{C}_k = \mathcal{C}_T^k$ being a function of k. In what follows we do not try to select the ideal value for the threshold parameter but we try to select a suitable model out of all $\mathcal{C}_{k=0}^K$ by minimizing certain criterion, which we will call 'strengthened Schwarz information criterion' (sSIC).

For any candidate $\mathcal{C}$ we define $\hat{f}_t^k$ the estimation of f_t that satisfies $\hat{f}_t^k = (\hat{\eta}_{i+1} - \hat{\eta}_i)^{-1} \sum_{j=\hat{\eta}_i+1}^{\hat{\eta}_{i+1}} X_j$ for $\hat{\eta}_i + 1 \leq t \leq \hat{\eta}_{i+1}$. By standard we denote $\hat{\sigma}_k^2 = 1/T \sum_{t=1}^T (X_T - \hat{f}_t^k)^2$ being the maximum likelihood estimator of residual variance. Further we define

$$sSCI(k) = \frac{T}{2} \log \hat{\sigma}^2 + k \log^\alpha T$$

For $\alpha = 1$ we get standard SIC penalty, used for example in Yao [1988] in the context of full penalized least-squares minimization.

Theorem 3 (Strengthened Schwarz Information). *Let X_T follow model* (2.1) *and suppose our assumptions* (3.2) *holds. Let N and $\eta_1, ..., \eta_N$ denote, respectively, the number and locations of change-points. Let $N \leq K$ where K is a certain constant independent of T. Let the constant $\alpha > 1$ be such that* $\log^\alpha T = o(\delta_T \underline{f}_T^2)$. *Let the candidate model $\{\mathcal{C}_k\}_{k=0}^K$ be produced by the WBS algorithm, and let $\hat{N} = arg \min_{k=0,1,...,K} sSCI(k)$. Then $P(\mathcal{A}_T) \geq 1 - C_1 T^{-1}(1 - \delta_T^2 T^{-2}/9)^M$, where*

$$\mathcal{A}_\mathcal{T} = \{\hat{N} = N; \max_{i=1,...,N} |\hat{\eta}_i - \eta_i| \leq C \log T (\underline{f}_T)^{-2}\}$$

for certain positive constants C, C_1.

Proof of this theorem can be found in the appendix of Fryzlewicz [2014].

The only parameter in this approach is α. We require $\alpha > 1$ to be stronger penalty than in the case of $\alpha = 1$, thus 'strengthened' SIC.

It is important to say that in this approach, unlike thresholding, the minimization of sSIC is independent of $\mathrm{Var}(\varepsilon_t)$ because of our logarithmic transformation of $\hat{\sigma}^2$. This makes $\mathrm{Var}(\varepsilon_t)$ have an additive impact on sSIC, thus having no impact on minimization.

The benefit of sSIC is that it is hopefully easier to find parameter of this procedure rather then finding good threshold parameter. On the other hand, it requires that $N \leq K$ for some finite K and the lowest admissible $\delta_t \underline{f}_T^2$ is larger than in the threshold approach. The requirement on finite K is common among penalized approaches in multiple change-point detection.

Example

Example (5). In this example, we will compare results obtained in Example 4 by WBS (threshold approach) with WBS utilizing sSIC criterion. We consider the same data as in example 4. Using sSIC penalty approach, we first choose k, the upper bound on the number of change-points we wish to detect. Then we find candidate models $\{\mathcal{C}_k\}_{k=0}^{K}$ using WBS algorithm. We then choose that model that minimizes the $sSIC(k)$. Since it would take so much time to display the algorithm step by step, we use the R package 'wbs' by Baranowski and Fryzlewicz [2019] to do the work for us. Reconstruction of this example with additional information can be found in the appendix of this work (A).

Remark when we use data from example 4 (shown in Figure (3.8) and (3.9)), the classic WBS algorithm tries its best and finds 5 change-points as shown in the example itself. We now try to get new change-point locations estimation using sSIC criterion. This approach is in the R package called 'wbs' done by using the function *wbs*(), but with an additional argument that chooses the type of penalty we want to use. After calling such a function on our data, we get the following result. Note that the reconstruction and detailed guide to how to obtain all the important information is shown in Appendix (A).

```
     ssic.penalty
[1]  45.29585  41.20883  39.85395  32.34140  24.71272
[2]  21.46938  20.78864  20.93993  25.58146  29.33684
```

This table shows the evaluated sSIC criterion for each candidate model from $\{\mathcal{C}_k\}_{k=0}^{K}$ starting on line [1] going from left to right, where the first line of the code ends with value for a specific model, that being $\mathcal{C}_4$. Second line continues with $\mathcal{C}_5$ and ends with $\mathcal{C}_8$.

Penalty magnitude decreases for increasing k to the point where k reaches 6 and then again starts to rise for increasing k. This results in having $k = 6$ minimizing the needed criterion. Recalling the table from example 4,

```
       s   e cpt       CUSUM      min.th scale   abs_cusum
 [1,]   1 300 163 -7.778865287 7.778865287     1 7.778865287
 [2,] 164 300 179  7.048670401 7.048670401     2 7.048670401
```

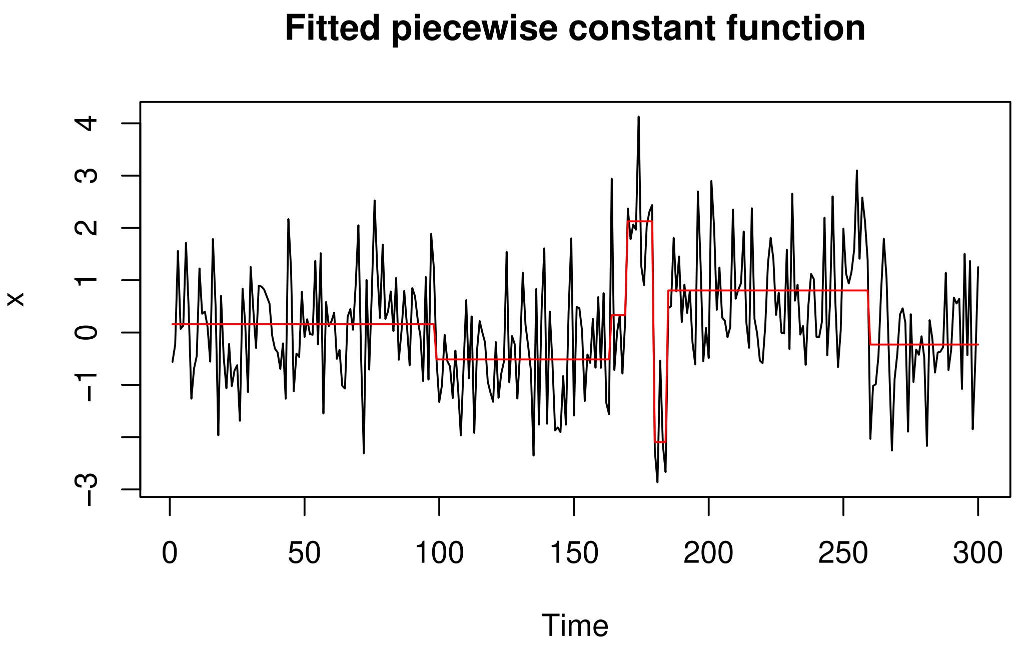

Figure 3.11: Time series (black line) for example 5 with $\hat{f}_t$ (red line) using WBS with sSIC criterion

```
  [3,] 180 300 184 -6.122027455 6.122027455     3 6.122027455
  [4,] 185 300 259  5.793638369 5.793638369     4 5.793638369
  [5,]   1 163  98  4.619866795 4.619866795     2 4.619866795
  [6,] 164 179 169 -4.048986994 4.048986994     3 4.048986994
  [7,] 185 259 249 -3.346755352 3.346755352     5 3.346755352
```

we now consider the first 6 change-points estimates to be 'good enough' instead of only 5 as in example 4. This gives us true f_t estimation shown in Figure (3.11).

4. Real Data Example

In this chapter we use both methods, BS and WBS, to analyze trends in daily logarithmic returns of Zoom Video Communications (ZOOM) company stock. We obtain the data with R package 'quantmod' by Ulrich [2007]. The time series consists of 1184 values starting on 18-04-2019 and ending on 31-12-2023. Data are visualized via R command *plot*() in Figure (4.1). This chapter shows the performance of standard BS and WBS with various parameter choices as well as wild binary segmentation with sSIC. In what follows we consider parameter C to have two possible values, that being $C = 1.3$ or $C = 1$.

4.1 Data Analysis

This section shows the performance of the following algorithms. At first, we assume BS with $C = 1$ and BS with $C = 1.3$. Then we upgrade the method and use WBS with the same constant C as for BS with a random interval approach and $M = 5000$. In the end, we analyse the data using sSIC along with WBS.

Standard Binary Segmentation results

We start with the 'simplest' method introduced in our text, that being standard binary segmentation. We are still using the R package called 'wbs' by Baranowski and Fryzlewicz [2019].

For $C = 1.3$ the BS returns an empty table for change-point estimations as well as constant $\hat{f}_t$. Figure (4.2) shows the result of the change-point analysis. After further investigation, we can see that in the first step, the BS algorithm evaluates all CUSUM statistic magnitudes $\left|\tilde{X}^b_{1,1184}\right|$. Its maximum occurs at $T = 380$ with CUSUM statistic magnitude $\left|\tilde{X}^{380}_{1,1184}\right| \doteq 0.1329$. Unfortunately, for $C = 1.3$, we get $\zeta_t \doteq 0.1459$ resulting in detecting no change-points in the first iteration of the algorithm since $\left|\tilde{X}^{380}_{1,1184}\right| < \zeta_t$. After decreasing the C parameter value to 1 we obtain different result.

For $C = 1$, we have $\zeta_t \doteq 0.1122$ allowing BS to detect change-point at $T = 380$, although no additional change-points are detected. We narrowed the Figure (4.3) to show the graph from $T = 340$ to $T = 420$ for better visualization of $\hat{f}_t$ made by BS with $C = 1$. Note that time index $T = 380$ corresponds with 19-10-2020. Detection of change-point at such a date could refer to the ongoing effects of the COVID-19 pandemic, being very close to the beginning of the school year or university semester.

Wild Binary Segmentation results

Moving on to WBS, we use the same constant C values as in the section with standard BS. With $C = 1.3$ giving us $\zeta_t \doteq 0.1459$ we have 20 change-point estimations. They occur at times 347, 346, 394, 396, 233, 234, 597, 844, 843, 36, 598, 409, 235, 34, 410, 656, 657, 901, 351, 899. We can see those change-point estimations with corresponding $\hat{f}_t$ in Figure (4.4) The change-points estimated between

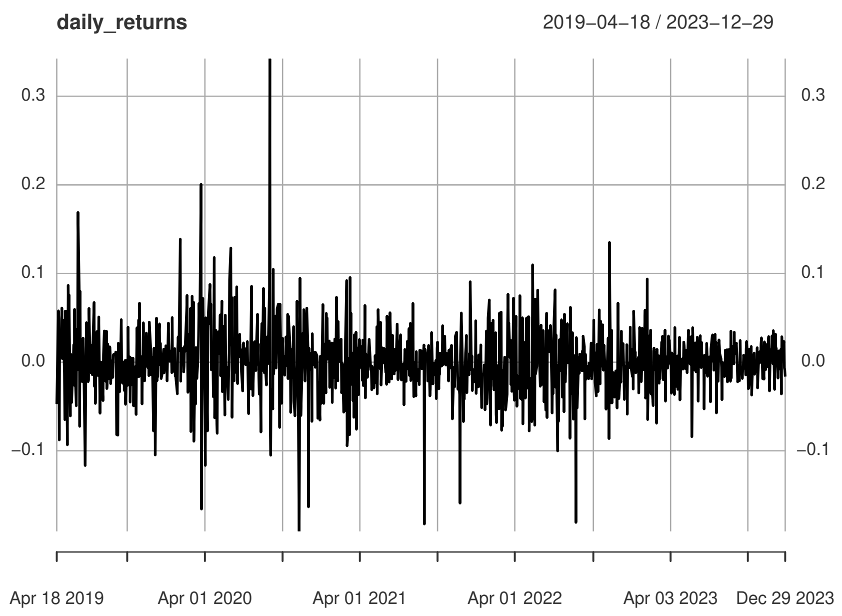

Figure 4.1: Logarithmic daily returns of ZOOM stock (black line)

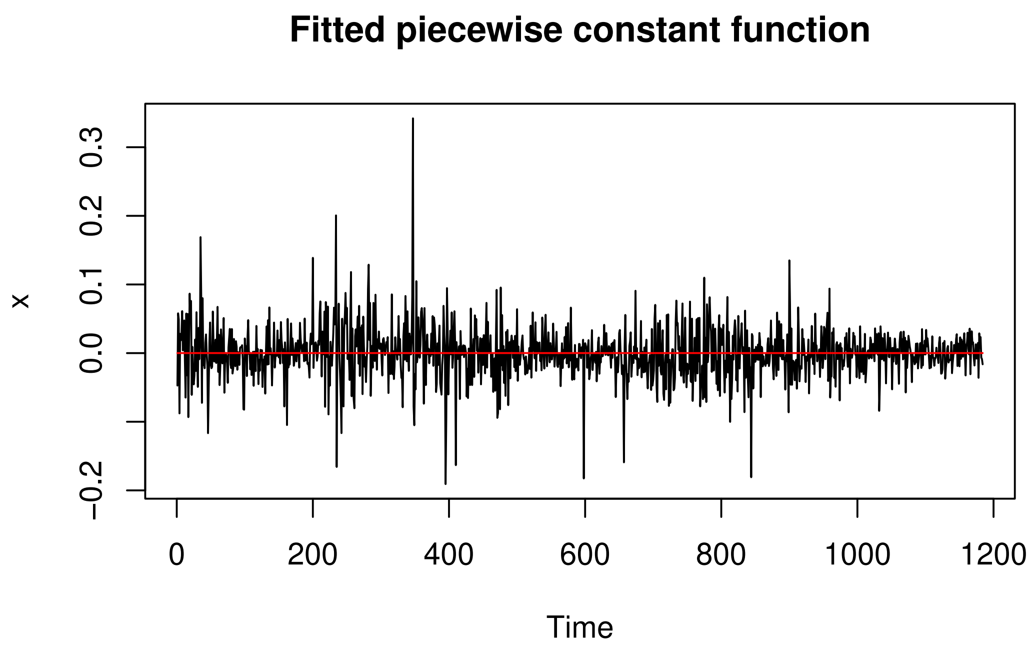

Figure 4.2: $\hat{f}_t$ (red line) made by standard BS with C = 1.3 for the logarithmic daily returns (black line)

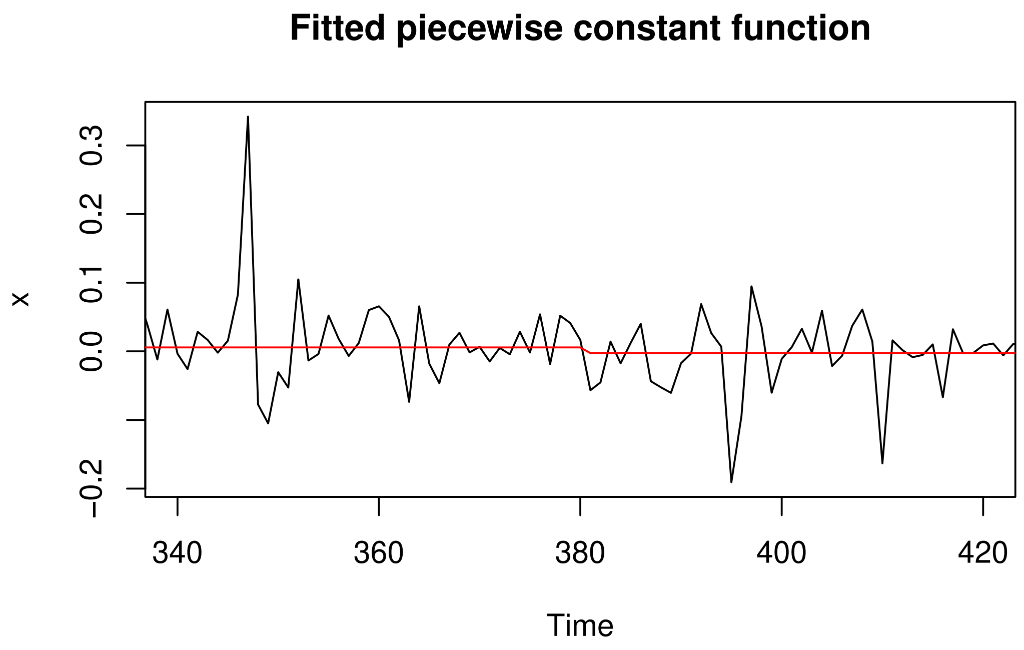

Figure 4.3: $\hat{f}_t$ (red line) made by standard BS with C = 1 for the logarithmic daily returns (black line) from $T = 340$ to $T = 420$

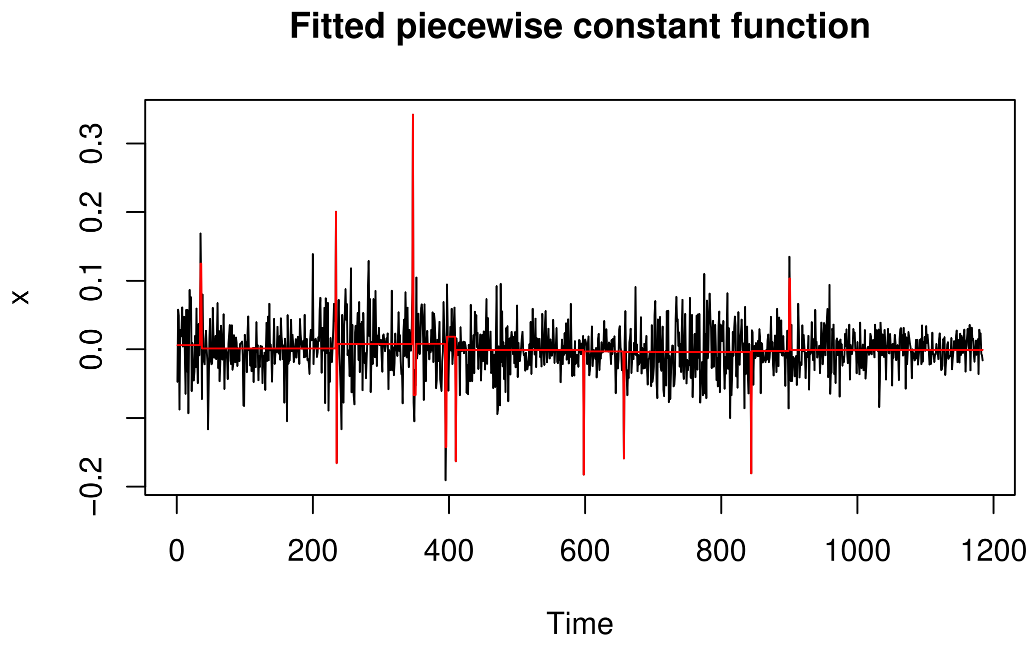

Figure 4.4: $\hat{f}_t$ (red line) made by WBS with C = 1.3 for the logarithmic daily returns (black line)

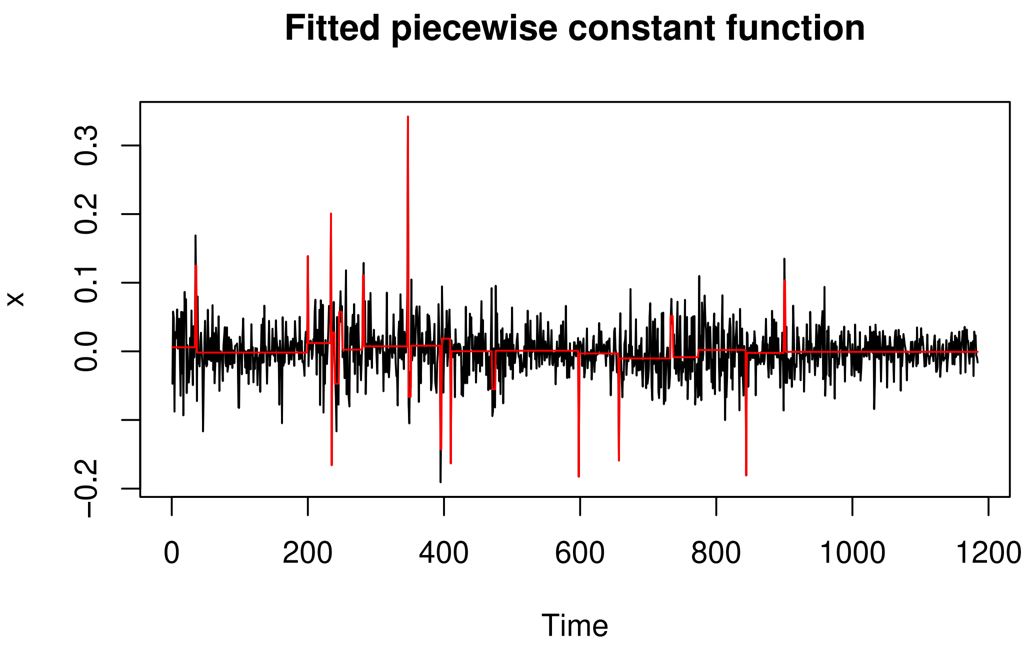

Figure 4.5: $\hat{f}_t$ (red line) made by WBS with C = 1 for the logarithmic daily returns (black line)

$T = 200$ and $T = 600$ can be attributed to the ongoing effects of the COVID-19 pandemic, which prompted many companies to transfer their operations from offices to virtual platforms such as Zoom. Important to say that the time of such transfers was mainly influenced by government policies in each country, introducing new rules for example for schools, companies, shops, etc. Additionally, the pandemic had a high impact on the economy, resulting also in stock market price fluctuations. Change-points detected somewhere around $T = 850$ and beyond could be the result of overall improving COVID-19 situation, where many companies and schools have already went 'back to normal'.

Very important to say, as mentioned in examples of this work, for different instances of the R program, we might get different random draws for WBS with $M = 5000$ resulting in slightly different change-point location estimations or different $\hat{N}$.

Now for $C = 1$ we get $\zeta_t \doteq 0.1122$ we have 32 change-point estimations at times 347, 346, 394, 396, 234, 233, 597, 598, 844, 34, 843, 409, 36, 235, 410, 901, 656, 657, 899, 282, 351, 199, 475, 280, 773, 470, 200, 732, 245, 239, 736, 250 shown in Figure (4.5). Additional change-point estimations, when compared to WBS with $C = 1.3$, could be also explained by COVID-19, which had an additional high impact on the economic situation in the world or with the schools and companies trying to adapt current pandemic situation. Change-points found after time index $T = 600$, that being 02-09-2021 and beyond, may align with the overall improvement in the pandemic situation and the decreased reliance on computers and online platforms as the only means of safe communication.

Now considering sSIC with $K = 50$, we get 6 change-points estimations at times 347, 346, 396, 394, 233, 234.

Method	$\hat{N}$
BS (C = 1.3)	0
BS (C = 1.0)	1
WBS (C = 1.3)	20
WBS (C = 1.0)	33
WBS (sSIC)	6

Table 4.1: Number of change-point estimations $\hat{N}$ for each method used in real data analysis

Change-point estimations found at time indices 34 and 36, corresponding with 06-06-2019 and 10-06-2019 cannot be explained with COVID-19, since it is too early. Those change-points could be explained as a result of ZOOM becoming a public company in April 2019 being very lucrative for investors. Other change-point estimations can be explained as above.

When we compare all methods in a simple Table (4.1), we have that the most change-point estimations we got by WBS with $C = 1$ and 0 change-point estimations with BS for $C = 1.3$. Since there is no right or wrong answer, we cannot choose the 'best' method in the sense of accuracy. Although we can say that WBS, being hopefully a more robust method than standard BS, could be considered slightly more reliable because it is able to detect change-points that are 'hidden' behind unfavorable change-point configuration.

A. Appendix

Example 1

We use the R package ’wbs’ made by Baranowski and Fryzlewicz [2019]. There we can find all the important information and details concerning R implementation.

Data construction:

```
data <- rnorm(300) + c(rep(1,50),rep(0,250))
```

We generate our data by defining the step function with step in t=50 and magnitude 1. Then we add standardized Gaussian noise, that is, with mean $\mu = 0$ and standard deviation $\sigma = 1$ resulting in the function shown in Figure (3.1).

We then call a function ’sbs’ on our data set.

```
s <- sbs(data)
```

To variable ’s’ we save important information. To extract the table shown in Example (3.2) we use a new variable called ’table’.

```
table <- s[[’res’]]
```

To create a visual image of our data and our true f_t, we can use

```
plot(data, type = ’l’, xlab = ’Time’, ylab = ’x’,
    main = "Time series for Example 1", ylim = c(-3, 4))

plot(c(rep(1,50),rep(0,250)), type = ’l’, xlab = ’time’,
    ylab = ’x’, main = "True f_t function for Example 1",
    ylim = c(-3, 4))
```

To sort the ’res’ table by the magnitude of CUSUM statistic value, but in absolute value, can simply be achieved by

```
abs_cusum = abs(table[,4])
table2 = cbind(table, abs_cusum)
sorted_table <- table2[order(table2[, "abs_cusum"],
    decreasing = TRUE), ]
```

We created a new column called ’abs_cusum’ as a transformation of the fourth column from the dataset called ’table’ and by using the function abs(), we get a transformation of CUSUM values to the corresponding absolute values. Argument ’decreasing = TRUE’ sorts the table by the last column, that being ’abs_cusum’, in decreasing order.

Example 2

Data for Example 2 are constructed like this:

```
    set.seed(123)
    segment1 <- rnorm(120, mean = 0, sd = 1)
    segment2 <- rnorm(20, mean = 1, sd = 1)
    segment3 <- rnorm(20, mean = 2.5, sd = 1)
    segment4 <- rnorm(140, mean = 0, sd = 1)
    time_series <- c(segment1, segment2, segment3, segment4)
```

This code results in step function f_t being 0 on interval $[1, 120]$, being 1 on $[121, 140]$, 2.5 on $[141, 160]$ and again being 0 on $[141, 300]$. The step function is later 'noised' by random numbers taken from a normal distribution with the mean equal to zero and variance equal to 1. The 'noised function', that being our data, we can illustrate via

```
    plot(time_series, type = "l",
        main = "Time Series for Example 2",
        xlab = "Time", ylab = "x", ylim = c(-3, 5))
```

resulting in plot shown in (3.4). We are able to extract the true function f_t with

```
    plot(ts(c(rep(0,120),rep(1,20),rep(2.5,20),rep(0,140))),
        ylim = c(-3,5), main = 'True f_t function for Example 2',
        ylab = 'x')
```

shown in figure (3.5).

Similarly to example 1, we use function *sbs*() to get the table shown in example 2.

```
    standard = sbs(time_series)
    standardtable <- standard[["res"]]
    abs_cusum = abs(standardtable[,4])
    standard2 = cbind(standardtable, abs_cusum)
    sorted_standard <- standard2[order(standard2[,
        'abs_cusum'], decreasing = TRUE), ]
```

This code sequence gives us all the information needed, and with

```
    plot(standard, ylim = c(-3, 5))
```

we get the true function estimation $\hat{f}_t$ shown in figure (3.6).

For the threshold value calculation, we need an absolute median estimation of σ:

```
    mad(time_series)
```

being

```
    [1] 1.006067.
```

Using the formula for ζ_t shown in example 2 results in

```
mad(time_series)*1.3*sqrt(2)*sqrt(log(300))
```

giving us the value for ζ_t

```
[1] 4.417399
```

Example 3

Here, identically to Example 2 (A) we reconstruct the data. The only difference here is the method we use. To use *wbs*() we do as follows.

```
wild = wbs(time_series, integrated = FALSE, m = 5000)
wild2 = wild[['res']]
abs_cusum = abs(wild2[,4])
wild2 = cbind(wild2, abs_cusum)
```

This gives us the table with sorted CUSUM by its absolute value shown in example 3. To plot the resulting $\hat{f}_t$ we use:

```
plot(wild, ylim = c(-3, 5))
```

and to obtain automatically evaluated ζ_t we type

```
w.cpt <- changepoints(wild)
w.cpt$th
```

which gives us a large table where almost at the beginning we can find

```
$th
[1] 4.012313
```

being the ζ_t value used in the threshold approach of the WBS algorithm.

Example 4

To reconstruct the data from example 4 we use a little bit more complicated code to be prepared for more sophisticated examples. To recreate Figure (3.8) we do as follows.

```
epsilon <- rnorm(300, mean = 0, sd = 1)
change_points <- c(1, 75, 80, 85, 170, 180, 185, 250,
    260, T)
magnitudes <- c(0, 1.5, 0.4, -0.3, 2.0, -1.8, 0.7, 1.5,
    -0.5, 0)
f_t <- rep(0, T)
for (i in 2:length(change_points)) {
    f_t[change_points[i - 1]:(change_points[i] - 1)]
    <- magnitudes[i - 1]
    }
```

This together gives:

```
X_t <- f_t + epsilon
```

By this code:

```
plot(X_t, type = "l", main = "Time series for Example 4",
    ylim = c(-3, 4), ylab = 'x', xlab= 'Time')

plot(f_t, type = "l", main = "True f_t function for example 4",
    ylim = c(-3, 4), ylab = 'x', xlab= 'Time')
```

we recreate exactly the plots as shown in (3.8) and (3.9). We call $wbs()$ on the data.

```
wbs_result_threshold <- wbs(X_t, M = 5000, integrated = FALSE)
```

the argument 'integrated = FALSE' is there to make sure we use WBS specifically and not the augmented version of the WBS algorithm. Further information can be found in Baranowski and Fryzlewicz [2019].

In what follows, we recreate the CUSUM table shown in example 4.

```
table1 = wbs_result_threshold[['res']]
abs_cusum = abs(table1[,4])
table2 = cbind(table1, abs_cusum)
sorted_table2 <- table2[order(table2[, 'abs_cusum'],
    decreasing = TRUE), ]
```

Note that for different executions of this code, we might after restarting the computer or restarting the program R obtain different configurations of random intervals used for WBS, thus resulting in slightly different CUSUM statistic values. This might make some change-points being ignored or some change-points might even be considered significant for constant ζ_t when executing the code multiple times in a row.

To plot the $\hat{f}_t$ that WBS makes, we do as follows.

```
w.cpt <- changepoints(wbs_result_threshold)
cps = w.cpt$cpt.th
```

and visualize the resulting $\hat{f}_t$ with

```
plot(wbs_result_threshold, unlist(cps), ylim = c(-3, 4))
```

for the plot corresponding to the case with the thresholding approach.

Example 5

Since in example 5 we use the same data as in example 4, we now only focus on the sSIC data extraction. Note that there is still some parameter choice to be done, that being the choice of parameter α. We choose $\alpha = 1.01$ to be slightly more strict than standard SIC but also close to the result obtainable by the standard SIC approach.

```
ssic = wbs(X_t, penalty = "ssic.penalty")
ssic_result <- changepoints(ssic)
```

After everything is done as shown above, we can type

```
ssic_result
```

which returns us a lot of useful information. Although we only need some of them.

```
ssic_result$cpt.ic$ssic.penalty
ssic_result$Kmax
ssic_result$ic.curve$ssic.penalty
ssic_result$no.cpt.ic
```

Since we did not set a specific value for the maximum number of estimated change-points k, the table shows us that it was automatically set to $k = 50$. Additionally, it shows us the 'curve' of sSIC criterion values for various k going from 0 to 50. And finally, in the bottom part, the table shows us locations of estimated change-points and their total number $\hat{N}$.

Conclusion

This work described two change-point detection methods called binary segmentation and wild binary segmentation. Those methods are trying to estimate the total number and the locations of structural changes in the data.

In the second chapter, we introduced our model and the theoretical problem. The following chapter introduced both algorithms with simple examples. We illustrated both algorithms using very easy-to-understand recursive steps as well as theorems that talk about the consistency of corresponding estimations. We also emphasized the importance of various parameter value choices and their influence on the result. Interesting approaches were introduced, that is, the threshold approach and the penalized approach achieved via strengthened Schwarz information criterion.

The last chapter showed the results of all methods with various constant value choices on a real data example. We were trying to find the number and locations of change-points in the logarithmic daily returns of the Zoom Video Communications stock. Binary segmentation showed up as a method that estimates fewer change-points than its competitor WBS. Change-point estimations correlated probably with the global political situation and economic situation, but most importantly with the pandemic caused by COVID-19. All methods were then compared with a simple table showing the number of estimated change-points, since we could not choose a winner as far as we had no right or wrong answer for this data analysis.

As shown in examples and real data analysis the wild binary segmentation method could be taken as a more robust method than classic binary segmentation since it can, unlike BS, detect change-points in more ’difficult configurations’ thanks to its ’narrowing’ feature.

Bibliography

Rafal Baranowski and Piotr Fryzlewicz. *wbs: Wild Binary Segmentation for Multiple Change-Point Detection*, 2019. URL `https://CRAN.R-project.org/package=wbs`. R package version 1.4, [Accessed 30-04-2024].

Rafal Baranowski, Yining Chen, and Piotr Fryzlewicz. Narrowest-over-threshold detection of multiple change points and change-point-like features. *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, 81(3):649–672, 2019. ISSN 1369-7412,1467-9868. doi: 10.1111/rssb.12322. URL `https://doi.org/10.1111/rssb.12322`.

Piotr Fryzlewicz. Wild binary segmentation for multiple change-point detection. *Ann. Statist.*, 42(6):2243–2281, 2014. ISSN 0090-5364,2168-8966. doi: 10.1214/14-AOS1245. URL `https://doi.org/10.1214/14-AOS1245`.

Piotr Fryzlewicz. Tail-greedy bottom-up data decompositions and fast multiple change-point detection. *Ann. Statist.*, 46(6B):3390–3421, 2018. ISSN 0090-5364,2168-8966. doi: 10.1214/17-AOS1662. URL `https://doi.org/10.1214/17-AOS1662`.

Joshua M. Ulrich. *quantmod: Quantitative Financial Modelling Framework*, 2007. URL `https://cran.r-project.org/web/packages/quantmod`. R package version 0.4.26, [Accessed 03-05-2024].

Yi-Ching Yao. Estimating the number of change-points via Schwarz' criterion. *Statist. Probab. Lett.*, 6(3):181–189, 1988. ISSN 0167-7152,1879-2103. doi: 10.1016/0167-7152(88)90118-6. URL `https://doi.org/10.1016/0167-7152(88)90118-6`.

List of Figures