Imperial College London

Department of Mathematics

Statistical Modeling for Streaming Data

Author: *Xing Liu*

Supervisor: *Dr Din-Houn Lau*

Date: August 6, 2018

Abstract

Implementing regression models to streaming data is a popular topic in many areas. This report involves developing online regressing and monitoring techniques, which can be useful for modeling streaming data in, e.g., civil and structure engineering.

In this project, we begin by introducing the recursive least-square algorithm, together with the iterative formulae for some important goodness-of-fit measures. We then consider the problem of isolated departures (outliers) by investigating the leverage and the studentized residual. Finally, we study various change-point detection methods and develop further on the Quandt's test, followed by their performance analysis on a simulated dataset. It turns out that this modified Quandt's test produces the most desirable empirical results.

I would like to thank Dr Lau for providing this interesting research opportunity and for his invaluable guidance throughout this project, without which I would not have been possible to complete this report. The content of this research is particularly enjoyable and I really appreciate it.

Contents

1 Introduction

Recall that an ordinary linear regression model under the normal theory assumption is one of the form

$$\underline{Y} = X\underline{\beta} + \underline{\epsilon}, \quad \text{where} \tag{1.1.1}$$

- $\underline{Y} \in \mathbb{R}^n$ is the response vector,
- $X \in \mathbb{R}^{n \times p}$ is a deterministic design matrix,
- $\underline{\beta} \in \mathbb{R}^p$ is the vector of coefficients,
- $\underline{\epsilon} \sim N(\underline{0}, \sigma I^{(n)})$ is the error term.

By simple algebra, the least square estimator of $\underline{\beta}$ is the solution of the normal equation

$$X^T X \underline{\hat{\beta}} = X^T \underline{Y}.$$

Under the assumption that X has full rank, this equation has a unique solution $\underline{\hat{\beta}} = (X^T X)^{-1} X^T \underline{Y}$, which minimizes the sum-of-errors function

$$S(\underline{\beta}) = |\underline{\epsilon}|^2 = |\underline{Y} - X\underline{\beta}|^2 = \sum_{i=1}^{n} (Y_i - \underline{X}_i^T \underline{\hat{\beta}})^2 = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2,$$

where $\underline{X}_i^T$ is the ith row of X, i = 1, ..., n. It is clear that to obtain the LSE, we need to compute the inverse of the matrix $X^T X$.

Now suppose we are interested in predicting the closing price at the GDP-USD exchange rate of the next second, using predictors such as the beginning, closing, highest/lowest price of the past 10 seconds, etc. Hence, in this case, a new set of data are being fed into our model per second, so $n = n(t)$ is varying with time. Theoretically, one could apply the above method every time a new data set $\underline{X}_{n+1}$ is obtained. However, as this algorithm involves inverting a matrix, it would be computationally expensive, especially when dealing with data updated at such a fast rate. To address this problem, a new algorithm called the Recursive Least-Square (RLS) [1] is presented in this report, which computes the least-square fit of $\underline{\beta}$ and other important measures in a recursive manner, thus speeding up the process.

2 Recursive Least-Square Algorithm

2.1 Normal Equation

The assumptions of the RLS on the model are very similar to that of the ordinary least-square algorithm. In this setting, the model is of the form:

$$\mathbb{E}[\underline{Y}(n)] = X(n)\underline{\beta}(n), \quad \text{where}$$

- $n = 1, 2, ...$ is the time variable,
- $\underline{Y}(n) = (Y_1, Y_2, ..., Y_n)^T \in \mathbb{R}^n$ is the response vector of varying length,
- $X(n) = \begin{pmatrix} \cdots & \underline{X}_1^T \cdots \\ \cdots & \underline{X}_2^T \cdots \\ & \vdots \\ \cdots & \underline{X}_n^T \cdots \end{pmatrix} \in \mathbb{R}^{n \times p}$ is the design matrix at time n,

- $\underline{\beta}(n) = (\beta_1(n), \beta_2(n), ..., \beta_p(n))^T \in \mathbb{R}^p$ is the vector of parameters that changes w.r.t time.

We aim to find a least-square fit $\hat{\underline{\beta}}(n)$ of $\underline{\beta}(n)$ that minimizes the weighted sum-of-errors:

$$S(\underline{\beta}(n)) := \left(\underline{Y}(n) - X(n)\underline{\beta}(n)\right)^T W(n) \left(\underline{Y}(n) - X(n)\underline{\beta}(n)\right), \tag{2.1.1}$$

where $W(n) \in \mathbb{R}^{n\times n}$ is a weighting matrix. In particular, we will focus on weighting matrices that are diagonal. Hence, **(2.1.1)** can be rewritten as

$$S(\underline{\beta}(n)) := \sum_{i=1}^{n} w_{ii}|Y_i - \underline{X}_i^T\underline{\beta}(n)|^2.$$

Although the choice of w_{ii} could be arbitrary, one popular setting is the **exponential forgetting factor**, which takes the form $w_{ii} = \lambda^{(n-i)}, \forall i = 1, ..., n$, where $\lambda \in (0, 1]$. Furthermore, in order to stabilize the design matrix to make it have full rank, an extra regularization term $\delta\lambda^n|\underline{\beta}(n)|^2$ is added to **(2.1.1)**. The sum-of-errors function then becomes

$$S(\underline{\beta}(n)) = \sum_{i=1}^{n} \lambda^{n-i}|Y_i - \underline{X}_i^T\underline{\beta}(n)|^2 + \delta\lambda^n|\underline{\beta}(n)|^2,$$

where $\delta \in \mathbb{R}^+$ is the regularization parameter. The LSE is then the unique $\hat{\underline{\beta}}(n)$ such that

$$\left.\frac{d}{d\beta_k}S(\underline{\beta}(n))\right|_{\beta_k=\hat{\beta}_k} = 0, \quad k = 1, ..., p.$$

Now, $\forall k = 1, ..., p$,

$$\begin{aligned}\frac{d}{d\beta_k}S(\underline{\beta}(n)) &= \frac{d}{d\beta_k}\left(\sum_{i=1}^{n}\lambda^{n-i}|Y_i - \underline{X}_i^T\underline{\beta}(n)|^2\right) + \frac{d}{d\beta_k}\left(\delta\lambda^n|\underline{\beta}(n)|^2\right)\\ &= -2\sum_{i=1}^{n}\left[\lambda^{n-i}X_{ik}(Y_i - \underline{\beta}^T\underline{X}_i)\right] + 2\delta\lambda^n\beta_k(n)\ .\end{aligned}$$

Setting $\left.\frac{d}{d\beta_k}S(\underline{\beta}(n))\right|_{\beta_k=\hat{\beta}_k} = 0$ gives

$$\begin{aligned} 0 &= -2\sum_{i=1}^{n}\lambda^{n-i}X_{ik}Y_i + 2\sum_{i=1}^{n}\lambda^{n-i}X_{ik}(\hat{\underline{\beta}}^T\underline{X}_i) + 2\delta\lambda^n\hat{\beta}_k(n)\\ \iff \sum_{i=1}^{n}\lambda^{n-i}X_{ik}Y_i &= \sum_{i=1}^{n}\lambda^{n-i}X_{ik}\left[\sum_{t=1}^{p}\hat{\beta}_tX_{it}\right] + \delta\lambda^n\hat{\beta}_k(n)\\ &= \sum_{i=1}^{n}\lambda^{n-i}X_{ik}\left[\sum_{t=1}^{p}\hat{\beta}_tX_{it} + \delta\lambda^n\hat{\beta}_k(n)\delta_{tk}\right]\\ &= \sum_{t=1}^{p}\left[\sum_{i=1}^{n}\lambda^{n-i}X_{ik}X_{it} + \delta\lambda^n\delta_{tk}\right]\hat{\beta}_t(n)\ ,\\ \iff [\underline{z}(n)]_k &= [D(n)\hat{\underline{\beta}}(n)]_k \quad, \forall k = 1, ..., p,\\ \iff D(n)\hat{\underline{\beta}}(n) &= \underline{z}(n) \quad , \end{aligned} \tag{2.1.2}$$

where δ_{tk} is the Kronecker delta function, and, $\forall k, t = 1, ..., p$,

- $[D(n)]_{kt} = \sum_{i=1}^{n} \lambda^{n-i} X_{ik} X_{it} + \delta\lambda^n \delta_{kt}$,
- $[\underline{z}(n)]_k = \sum_{i=1}^{n} \lambda^{n-i} X_{ik} Y_i$.

Therefore,

$$D(n) = \sum_{i=1}^{n} \lambda^{n-i} \underline{X}_i \underline{X}_i^T + \delta\lambda^n I^{(p)} \ , \tag{2.1.3}$$

$$\underline{z}(n) = \sum_{i=1}^{n} \lambda^{n-i} \underline{X}_i Y_i z. \tag{2.1.4}$$

Equation **(2.1.2)** is known as the normal equation for the weighted least-square model. If n is far larger than p and X is thus believed to have full rank, one could set $\delta = 0$ to neglect the regularization term. In this case, by defining matrix $W(n) \in \mathbb{R}^{n\times n}$ such that $W_{ij} := \lambda^{n-i}\delta_{ij}, \forall i,j = 1,...,n$, **(2.1.2)** is algebraically equivalent to

$$(X^T(n)W(n)X(n))\underline{\hat{\beta}}(n) = X^T(n)W(n)\underline{Y}(n) \ . \tag{2.1.5}$$

Therefore, on choosing δ such that $D(n)$ is non-singular, the least-square estimator is

$$\underline{\hat{\beta}}(n) = D^{-1}(n)\underline{z}(n) = (X^T(n)W(n)X(n))^{-1}X^T(n)W(n)\underline{Y}(n) \ .$$

2.2 Recursive Formulae for the Weighted LSE

Following the definitions given in the previous section, a recursive algorithm of computing the least-square fit can be developed. Firstly, isolating the $i = n$ term in **(2.1.3)** gives

$$\begin{aligned} D(n) &= \lambda \left[\sum_{i=1}^{n-1} \lambda^{n-1-i} \underline{X}_i \underline{X}_i^T + \delta\lambda^{n-1} I^p \right] + \underline{X}_n \underline{X}_n^T \\ &= \lambda D(n-1) + \underline{X}_n \underline{X}_n^T \ . \end{aligned}$$

Similar manipulation on **(2.1.4)** leads to

$$\begin{aligned} \underline{z}(n) &= \lambda \left[\sum_{i=1}^{n-1} \lambda^{n-1-i} \underline{X}_i Y_i \right] + \underline{X}_n Y_n \\ &= \lambda \underline{z}(n-1) + \underline{X}_n Y_n \ . \end{aligned}$$

Next, in order to find the inverse of $D(n)$ in a iterative manner, we make use of the following lemma [2].

Lemma 2.1 (*Matrix Inversion Lemma*)**.** *Suppose A and B are $p \times p$ positive definite matrices, and $A = B^{-1} + CD^{-1}C^T$, where C, D are $p \times n$, $n \times n$, respectively. Then,*

$$A^{-1} = B - BC(D + C^TBC)^{-1}C^TB$$

Proof. By direct multiplication of A to A^{-1}. □

Now, assume δ is chosen such that $D(n)$ is non-singular. Then it is also positive definite. By **Lemma 2.1** applied to $A = D(n), B^{-1} = \lambda D(n-1), C = \underline{X}_n, D = 1$, we have

$$D^{-1}(n) = \lambda^{-1}D^{-1}(n-1) - \frac{\lambda{-2}D^{-1}(n-1)\underline{X}_n\underline{X}_n^T D^{-1}(n-1)}{1 + \lambda^{-1}\underline{X}_n^T D^{-1}(n-1)\underline{X}_n} \ .$$

Write $\begin{cases} \widetilde{D}(n) := D^{-1}(n) \\ \underline{\gamma}(n) := \frac{\lambda^{-1}D^{-1}(n-1)\underline{X}_n}{1+\lambda^{-1}\underline{X}_n^T D^{-1}(n-1)\underline{X}_n} \end{cases}$,

then $\begin{cases} \widetilde{D}(n) = \lambda^{-1}\widetilde{D}(n-1) - \underline{\gamma}(n)\underline{X}_n^T\widetilde{D}(n-1) \ , \\ \underline{\gamma}(n) = \left[\lambda^{-1}\widetilde{D}(n-1) - \lambda^{-1}\underline{X}_n^T\widetilde{D}(n-1)\right]\underline{X}_n = \widetilde{D}(n)\underline{X}_n \ , \end{cases}$
where $\underline{\gamma}(n)$ is called the gain vector, i.e. the input data transformed by the inverse of the correlation matrix $D(n)$. Furthermore,

$$\begin{aligned} \underline{\hat{\beta}}(n) &= \widetilde{D}(n)\underline{z}(n) \\ &= \lambda\widetilde{D}(n)\underline{z}(n-1) + \widetilde{D}(n)\underline{X}_nY(n) \\ &= \widetilde{D}(n-1)\underline{z}(n-1) - \widetilde{D}(n-1)\underline{X}_n\underline{X}_n^T\widetilde{D}(n-1)\underline{z}(n-1) + \widetilde{D}(n)\underline{X}_nY(n) \\ &= \underline{\hat{\beta}}(n-1) - \underline{\gamma}(n)\underline{X}_n^T\underline{\hat{\beta}}(n-1) + \underline{\gamma}(n)Y(n) \\ &= \underline{\hat{\beta}}(n-1) + \underline{\gamma}(n)\left(Y(n) - \underline{X}_n^T\underline{\hat{\beta}}(n-1)\right) \\ &= \underline{\hat{\beta}}(n-1) + \underline{\gamma}(n)\zeta(n) \ , \end{aligned}$$

where $\zeta(n) := Y(n) - \underbrace{\underline{X}^T(n)\underline{\hat{\beta}}(n-1)}_{\text{a priori estimate of the response data based on the old LSE}}$.

To summarize, a flow chart of the RLS algorithm is shown in **Figure 1**.

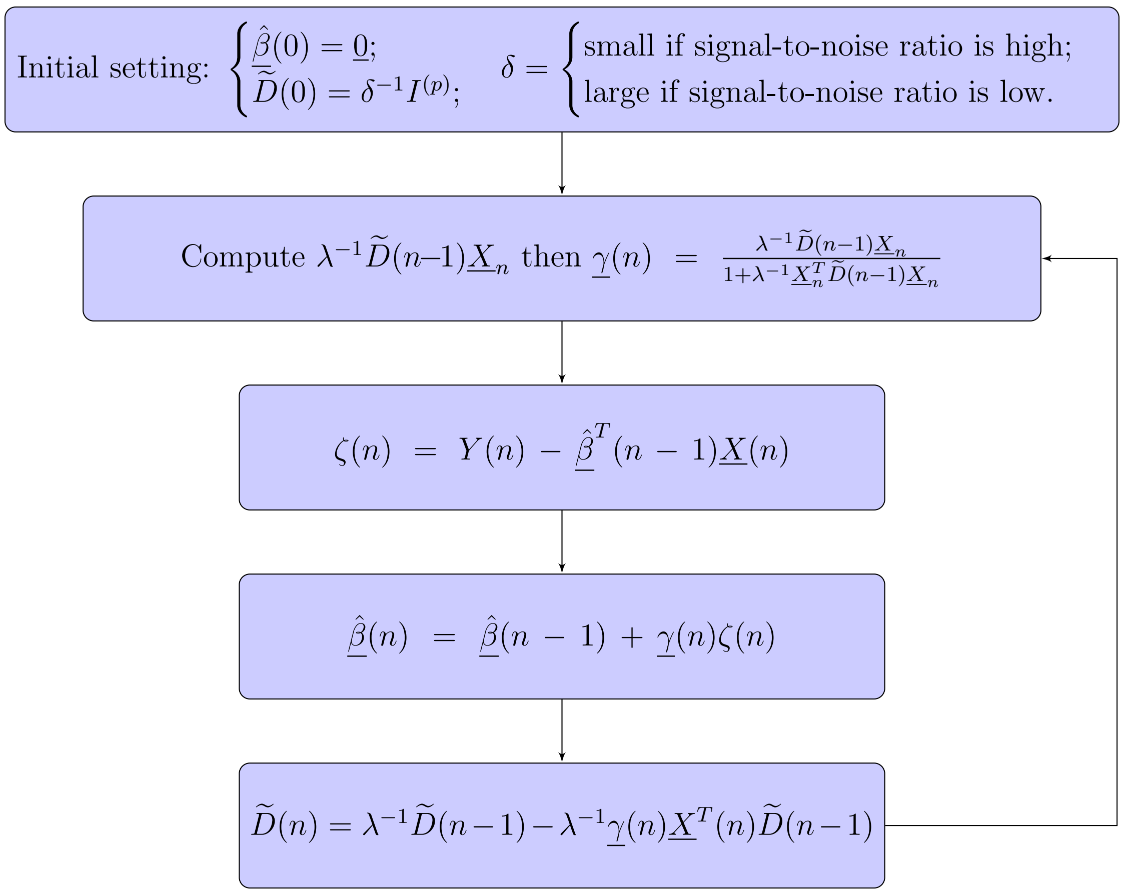

Initial setting: $\begin{cases} \underline{\hat{\beta}}(0) = \underline{0}; \\ \widetilde{D}(0) = \delta^{-1}I^{(p)}; \end{cases}$ $\delta = \begin{cases} \text{small if signal-to-noise ratio is high;} \\ \text{large if signal-to-noise ratio is low.} \end{cases}$
Compute $\lambda^{-1}\widetilde{D}(n–1)\underline{X}_n$ then $\underline{\gamma}(n) = \frac{\lambda^{-1}\widetilde{D}(n-1)\underline{X}_n}{1+\lambda^{-1}\underline{X}_n^T\widetilde{D}(n-1)\underline{X}_n}$
$\zeta(n) = Y(n) - \underline{\hat{\beta}}^T(n-1)\underline{X}(n)$
$\underline{\hat{\beta}}(n) = \underline{\hat{\beta}}(n-1) + \underline{\gamma}(n)\zeta(n)$
$\widetilde{D}(n) = \lambda^{-1}\widetilde{D}(n-1) - \lambda^{-1}\underline{\gamma}(n)\underline{X}^T(n)\widetilde{D}(n-1)$

Figure 1: *The RLS Algorithm.*

2.3 Recursive Computation for the Weighted RSS

Similar to the ordinary least-square case, various goodness-of-fit measures could be used in order to quantify how well the RLS model fits the real data. Amongst these, one common yet important measure is the residual sum of squares (**RSS**). Recall that, in the case of exponential forgetting factors, the weighted RSS is defined as

$$S(\underline{\hat{\beta}}(n)) := \left(\underline{Y}(n) - \underline{\hat{Y}}(n)\right)^T W(n) \left(\underline{Y}(n) - \underline{\hat{Y}}(n)\right) .$$

Note that

$$\begin{aligned}
\underline{Y}(n)^T W(n) \underline{Y}(n) &= \left|\underline{Y}(n) - \underline{\hat{Y}}(n) + \underline{\hat{Y}}(n)\right|^2_{W(n)} \\
&= |\underline{Y}(n)|^2_{W(n)} + \left|\underline{Y}(n) - \underline{\hat{Y}}(n)\right|^2_{W(n)} + 2\underbrace{\underline{\hat{Y}}^T(n) W(n) \left(\underline{Y}(n) - \underline{\hat{Y}}(n)\right)}_{=0 \text{ by orthogonality.}} \\
&= |\underline{Y}(n)|^2_{W(n)} + \left|\underline{Y}(n) - \underline{\hat{Y}}(n)\right|^2_{W(n)} .
\end{aligned}$$

That $\underline{\hat{Y}}^T(n) W(n) \left(\underline{Y}(n) - \underline{\hat{Y}}(n)\right) = 0$ follows directly by noticing that $\underline{\hat{\beta}}$ is the unique solution to the normal equation **(2.1.5)**. Now,

$$\begin{aligned}
S(\underline{\hat{\beta}}(n)) &= \left|\underline{Y}(n) - \underline{\hat{Y}}(n)\right|^2_{W(n)} = |\underline{Y}(n)|^2_{W(n)} - \left|\underline{\hat{Y}}(n)\right|^2_{W(n)} \\
&= |\underline{Y}(n)|^2_{W(n)} - \underline{Y}^T(n) W^T(n) X(n) D^{-1}(n) X^T(n) W(n) X(n) D^{-1}(n) X^T(n) W(n) \underline{Y}(n) \\
&= |\underline{Y}(n)|^2_{W(n)} - \left[\underline{Y}^T(n) W^T(n) X(n) D^{-1}(n)\right] D(n) \left[D^{-1}(n) X^T(n) W(n) \underline{Y}(n)\right] \\
&= |\underline{Y}(n)|^2_{W(n)} - \underline{\hat{\beta}}^T(n) D(n) \underline{\hat{\beta}}(n) \\
&= |\underline{Y}(n)|^2_{W(n)} - \underline{\hat{\beta}}^T(n) \underline{z}(n) \\
&= |\underline{Y}(n)|^2_{W(n)} - \underline{z}^T(n) D^{-1}(n) \underline{z}(n) \\
&= \lambda \left[\sum_{i=1}^{n-1} \lambda^{n-i-1} |Y_i|^2 + |Y_n|^2 - \underline{z}^T(n-1) \underline{\hat{\beta}}(n-1)\right] \\
&\quad + Y_n \left[Y_n - \underline{X}^T(n) \underline{\hat{\beta}}(n-1)\right] - \underline{z}^T(n) \underline{\gamma}(n) \zeta(n) \\
&= \lambda \left[\sum_{i=1}^{n-1} \lambda^{n-i-1} |Y_i|^2 + |Y_n|^2 - \underline{z}^T(n-1) \underline{\hat{\beta}}(n-1)\right] \\
&\quad + Y_n \zeta(n) - \underline{z}^T(n) \underline{\gamma}(n) \zeta(n) ,
\end{aligned} \tag{2.3.1}$$

where we have used that $D(n)$ is symmetric and substituted the iterative expressions for $\underline{z}(n)$ and $\underline{\hat{\beta}}(n)$, and that $\sum_{i=1}^{n} \lambda^{n-i} |Y_i|^2 = \lambda \sum_{i=1}^{n-1} \lambda^{n-i-1} |Y_i|^2 + |Y_n|^2$. By definition, the expression inside the first bracket equals to $S(\underline{\hat{\beta}}(n-1))$. Furthermore, the last term gives

$$\underline{z}^T(n) \underline{\gamma}(n) = \underline{z}^T(n) D^{-1}(n) \underline{X}_n = \left(D^{-1}(n) \underline{z}(n)\right)^T \underline{X}_n = \underline{\hat{\beta}}^T(n) \underline{X}_n ,$$

where we have again used the symmetry property of $D(n)$. Substituting this into **(2.3.1)**,

$$\begin{aligned}
S(\underline{\hat{\beta}}(n)) &= \lambda S(\underline{\hat{\beta}}(n-1)) + Y_n \zeta(n) - \underline{\hat{\beta}}(n) \underline{X}_n \zeta(n) \\
&= \lambda S(\underline{\hat{\beta}}(n-1)) + \zeta(n) e_n .
\end{aligned} \tag{2.3.2}$$

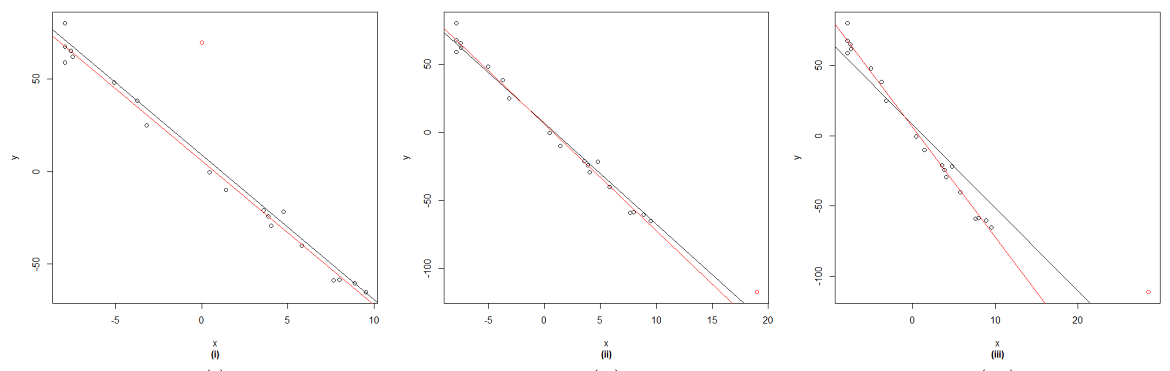
Hence, one can compute the weighted sum of squares iteratively using **(2.3.2)**. The RSS could be helpful in detecting outliers: if it suddenly increases by a large amount when a new piece of data is given, this data could be a potential outlier.

However, care must be taken when making such conclusions. For example, if the true beta vector varies with time and is different for the training and the testing model, then any data fed during the test phrase would be flagged (mistakingly) as “outliers”. This is where the forgetting factors will play a vital role. With simple modifications, the weighted RSS would still be helpful with making preliminary identifications on outliers, or a change in the true beta values. More on this topic can be found in the following section.

3 Outliers

Outliers are defined to be points for which the observations y_i are far from the values predicted by the model[3, p. 96]. They may occur in data for a wide range of reasons and could significantly affect the precision and the reliability of the model. As mentioned previously, in some cases, outlying observations may also indicate a change in the underlying model. In this chapter, some common goodness-of-fit measures in the RLS setting are presented.

In general, there are three types of configuration that are worth distinguishing [4], which are illustrated with a simulated dataset in **Figure 2**. The dataset consists of 20 samples, where one of them is modified in each case to represent an extreme observation. The first type is that of *leverage*, as shown in (i). Here, including the outlier does not significantly affect the gradient of the best-fit line, meaning that it has a small leverage compared with the other two cases. The second is that of *consistency*, which is demonstrated in (ii). This outlier, although having an extreme value, roughly follows the tendency of the fitted line. The third idea is termed as *influence*, as illustrated in (iii). Here, removing the outlier greatly affects the gradient of the fitted line, whereas this change is minor in (i) and (ii). In the rest of this section, we will explore some typical measures of these configurations.



(i) (ii) (iii)

Figure 2: *Plots showing the various effect that outliers may cause to the best fit line. Red points: outliers; black lines: best-fit lines; red lines: best-fit lines excluding the outliers.*

3.1 Leverage

We have seen in the previous chapter that $\underline{\hat{Y}} = X\left(X^TWX\right)^{-1}X^TW\underline{Y}$, and $H := X\left(X^TWX\right)^{-1}X$ is defined to be the hat matrix. One well-known measure of *leverage* [4] is defined to be h_{ii}, the ith diagonal entry of H, i.e.

$$h_{ii} = \underline{X}_i^T(X^TWX)^{-1}\underline{X}_i = \underline{X}_i^T\widetilde{D}\underline{X}_i \ .$$

It is clear that H is symmetric and idempotent in the sense that $HWH = H$. By direct algebraic manipulations considering its idempotent property, one has

$$0 \leq h_{ii} \leq 1 \quad \forall i.$$

Furthermore, note that the variance of the ith error term is $var(e_i) = \sigma^2(1 - h_{ii})$. Hence, if $h_{ii} \approx 1$, then y_i has high leverage, and $var(e_i)$ is low, which means that the design matrix is forcing the model fit to be good at the covariates of the ith observation. On the other hand, if $h_{ii} \approx 0$, then the fit is less affected by the ith observation [5]. Therefore, one should take notice when the leverage is high. A rule of thumb suggested by Hoaglin and Welsch in 1978 [5] is to pay attention when $h_{ii} > 2r/n$, where $r = rank(X)$.

For an interpretation of the leverages [5], note that adding an extra term ΔY_i to Y_i gives

$$\hat{Y}_i + \Delta\hat{Y}_i = \underline{X}_i^T (X^T W X)^{-1} \underline{X}_i W_{ii} \left(Y_i + \Delta Y_i\right) = \underbrace{H_{ii} W_{ii} Y_i}_{=\hat{Y}_i} + H_{ii} W_{ii} \Delta Y_i$$

$$\implies \Delta\hat{Y}_i = \underbrace{H_{ii}}_{\text{change in } \hat{Y}_i \text{ per unit change in } Y_i} W_{ii} \Delta Y_i \ .$$

To demonstrate this, three plots of the leverages against the fitted values are shown in **Figure 3**, corresponding to the three datasets used in **Figure 2**, respectively. Obviously, the second and the third graphs show strong evidence that the red dots are potential outliers. However, in the first plot, despite of being an true outlier, the red dot only has a low leverage. This is not surprising given the plots in **Figure 2**, as including the outlier in the first plot barely changes the gradient of the best-fit line.

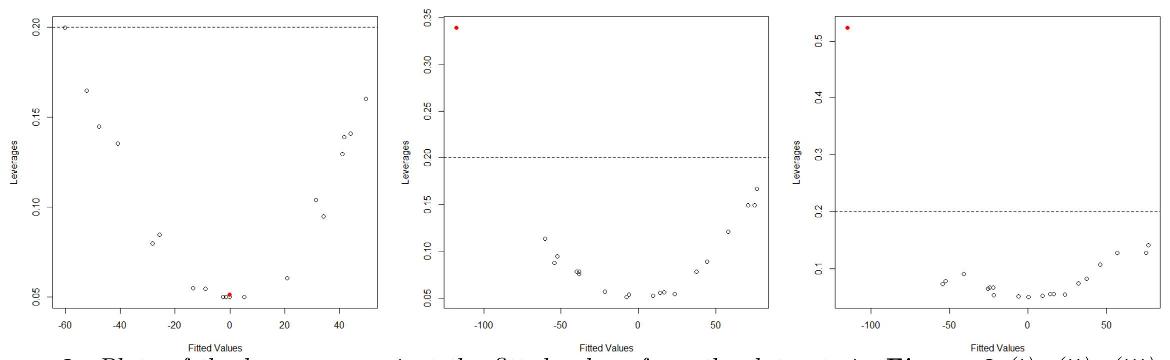


Figure 3: *Plots of the leverages against the fitted values from the datasets in* ***Figure 2*** *(i), (ii), (iii) (left to right), respectively.*

3.2 The Standardized and Studentized Residuals

From the model **(1.1.1)**, it is obvious that the residual associated with the ith observation follows a normal distribution $e_i \sim N(0, \sigma^2(1 - h_{ii}))$. A residual is called standardized if it is normalized to have constant variance [4, p. 396]. In particular, the standardized residuals is defined as

$$r_i := \frac{e_i}{\sqrt{1 - h_{ii}}} \ .$$

If r_i is then normalized by $\hat{\sigma}$, then we call it the *studentized* residual, i.e.

$$t_i := \frac{r_i}{\hat{\sigma}} = \frac{e_i}{\sqrt{\hat{\sigma}^2 \left(1 - h_i\right)}} \sim N(0, 1) \ .$$

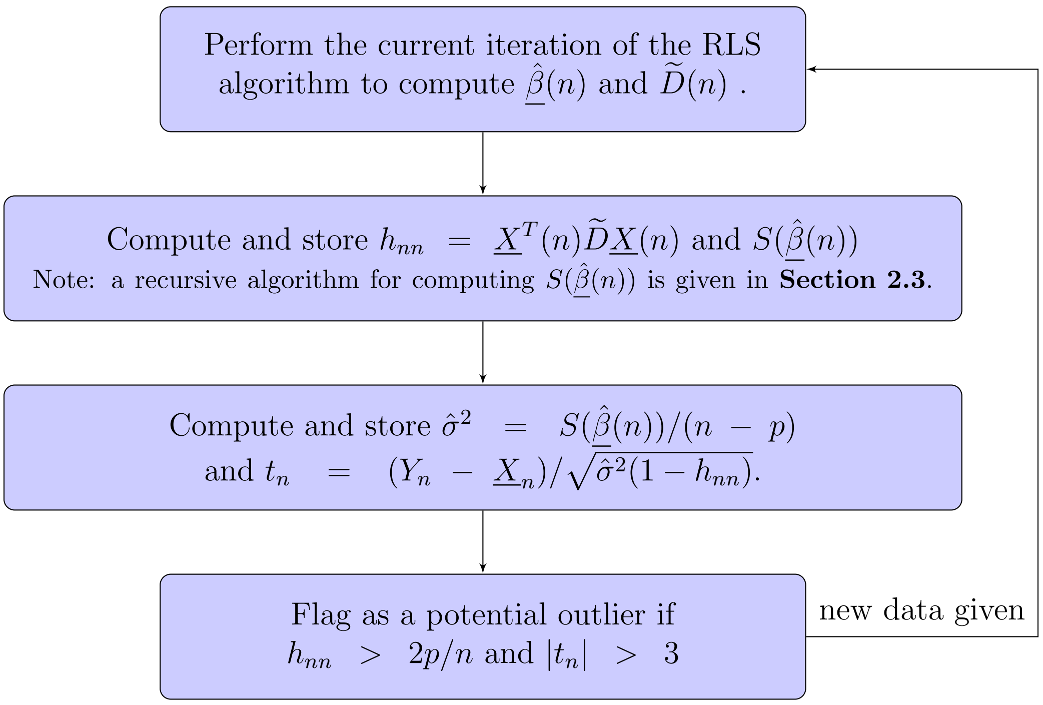

Perform the current iteration of the RLS algorithm to compute $\underset{\sim}{\hat{\beta}}(n)$ and $\widetilde{D}(n)$.
Compute and store $h_{nn} = \underset{\sim}{X}^T(n)\widetilde{D}\underset{\sim}{X}(n)$ and $S(\underset{\sim}{\hat{\beta}}(n))$
Note: a recursive algorithm for computing $S(\underset{\sim}{\hat{\beta}}(n))$ is given in **Section 2.3**.
Compute and store $\hat{\sigma}^2 = S(\underset{\sim}{\hat{\beta}}(n))/(n - p)$ and $t_n = (Y_n - \underset{\sim}{X}_n)/\sqrt{\hat{\sigma}^2(1-h_{nn})}$.
Flag as a potential outlier if $h_{nn} > 2p/n$ and $|t_n| > 3$
new data given

Figure 4: *Flagging potential outliers.*

Therefore, any observations with extreme values of studentized residuals can be potential outliers, typically outside of the range $[-3, 3]$.

Figure 5 and **Figure 6** compares the studentized residuals of the dataset (i) and (iii) in **Figure 2** against the fitted values and leverages, respectively. Note that in the second plot of **Figure 6**, the red dot has both high leverage and studentized residual. This is a particularly strong evidence of being an outlier. Hence, when performing the RLS algorithm, a natural way of detecting potential outliers is to flag the observations whose leverage is greater than $2p/n$ and absolute value of studentized residual greater than 3, as shown in the flowchart **Figure 5**.

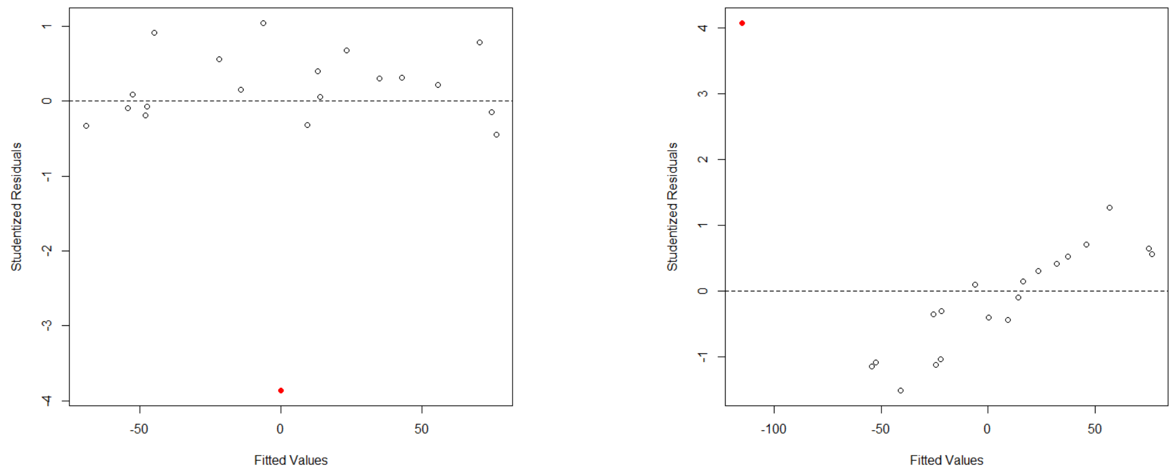


Figure 5: *Plots of the studentized residuals against the fitted values from the datasets in* ***Figure 2*** *(i), (iii), respectively.*

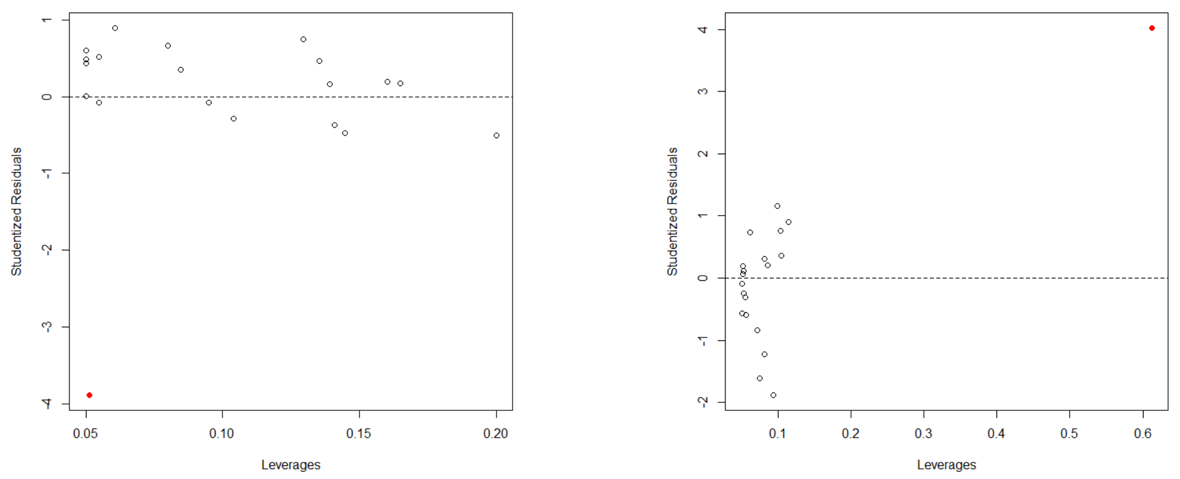


Figure 6: *Plots of the leverages against the studentized residuals from the datasets in* ***Figure 2*** *(i), (iii), respectively.*

4 Change-Point Detection

This section considers the situations where there is a change in the underlying model at some specific time point τ. This could be a change in the parameters ($\underline{\beta}$ or σ^2 in the ordinary linear model case) or in the model itself, and τ may or may not be known a priori. Putting this in a statistical setting, suppose that τ is an actual change point. We are interested in testing the hypothesis

$$H_0^{(\tau)} : M_0(\tau) \overset{d}{=} M_1(\tau) \quad \textit{against} \quad H_1^{(\tau)} : M_0(\tau) \overset{d}{\neq} M_1(\tau) \ ,$$

where $M_0(\tau)$ and $M_1(\tau)$ are the models before and after τ, respectively.

In general, change-point detections may involve estimation of several quantities: $(i)$the number of change points, $(ii)$the change points themselves and $(iii)$the models before and after the change points. However, since only time-series data is of interest in this report, we can assume without loss of generality that only 1 change point is involved. In this section, a set of change-point detection methods will be presented, followed by some applications and performance analyses on stimulated data.

4.1 The Quandt's Method with Moving Average Band

The Quandt's Method, proposed by Richard Quandt [6, 7], involves a hypothesis test on the regime of the linear model before and after a specific time point using the loglikelihood ratio. Here, we develop further on this test by making use of the *moving average band*, which will be defined later in this section, as reference values. More formally, consider the hypothesis test

$$H_0^{(\tau)} : \underline{\beta}_0 = \underline{\beta}_1 \quad \textit{against} \quad H_1^{(\tau)} : \underline{\beta}_0 \neq \underline{\beta}_1 \ .$$

Hence, $\underline{Y} \sim N(X\underline{\beta}_0, \sigma_0^2)$ under $H_0^{(\tau)}$, and $\underline{Y}_0^{(\tau)} \sim N(X_0^{(\tau)}\underline{\beta}_0, \sigma_0^2)$, $\underline{Y}_1^{(\tau)} \sim N(X_1^{(\tau)}\underline{\beta}_1, \sigma_1^2)$ under $H_1^{(\tau)}$, where $\underline{Y}_k^{(\tau)}$ and $X_k^{(\tau)}$ are the response vector and the design matrix consisting data before

$(k = 0)$ and after $(k = 1)$ time τ, respectively, and σ_0^2, σ_1^2 may or may not be known. In the former case, the test statistic is

$$\begin{aligned}\hat{R}_\tau &:= \log \frac{L(\hat{\underline{\beta}}_0|\underline{x}_0, \sigma_0^2)L(\hat{\underline{\beta}}_1|\underline{x}_1, \sigma_1^2)}{L(\hat{\underline{\beta}}_0|\underline{x}_0, \sigma_0^2)L(\hat{\underline{\beta}}_0|\underline{x}_0, \sigma_1^2)} \\ &= \log \frac{L(\hat{\underline{\beta}}_1|\underline{x}_1, \hat{\sigma}_1^2)}{L(\hat{\underline{\beta}}_0|\underline{x}_0, \hat{\sigma}_1^2)} ,\end{aligned}$$

where $L(\cdot)$ is the likelihood function and $\hat{\underline{\beta}}_k$ are MLE's (thus also the LSE's) of the true parameters $\underline{\beta_k}$ based on some reference and test sample, respectively, for $k = 0, 1$.

However, in practice, the variance is most likely to be unknown. In this case, it needs to be estimated using the MLE and the test statistic then becomes

$$\hat{R}_\tau = \log \frac{L(\hat{\underline{\beta}}_1, \hat{\sigma}_1^2|\underline{x}_1)}{L(\hat{\underline{\beta}}_0, \hat{\sigma}_1^2|\underline{x}_0)} .$$

Note that the parameter space under H_0 is always contained in the parameter space under H_1. This means that the loglikelihood ratio is always greater than 1, giving that $\hat{R}_\tau \geq 0$ with $\hat{R}_\tau \approx 0$ being very likely if H_0 was true. Hence, H_0 is rejected for $\hat{R}_\tau$ larger than some (positive) threshold, say, h.

Although this test method has been widely studied and applied by many researchers, it was pointed out [8] that the asymptotic distribution of $\hat{R}_\tau$ is unknown. Therefore, h has to be chosen 'case by case'. Another question yet to be answered is the choice of the reference and the test sample. These two questions shall be discussed in detail in the rest of this section.

For conciseness, we shall refer the sizes of the reference and the test sample as n_{ref} and n_{tes}, and the index of the starting time points of these samples as t_{ref} and t_{tes} (which are greater than p to ensure full rank), respectively. Since time-series data is of interest in this report, it is natural to select the reference and the test sample recursively, i.e. at each time point, the oldest datum in each of the two samples are forgotten while a new datum is added, as illustrated in **Figure 7**. This method of sampling is adopted from Kawahara [9] and is slightly modified. Note that, to eliminate correlation, the samples are chosen such that $t_{tes} = t_{ref} + n_{ref}$.

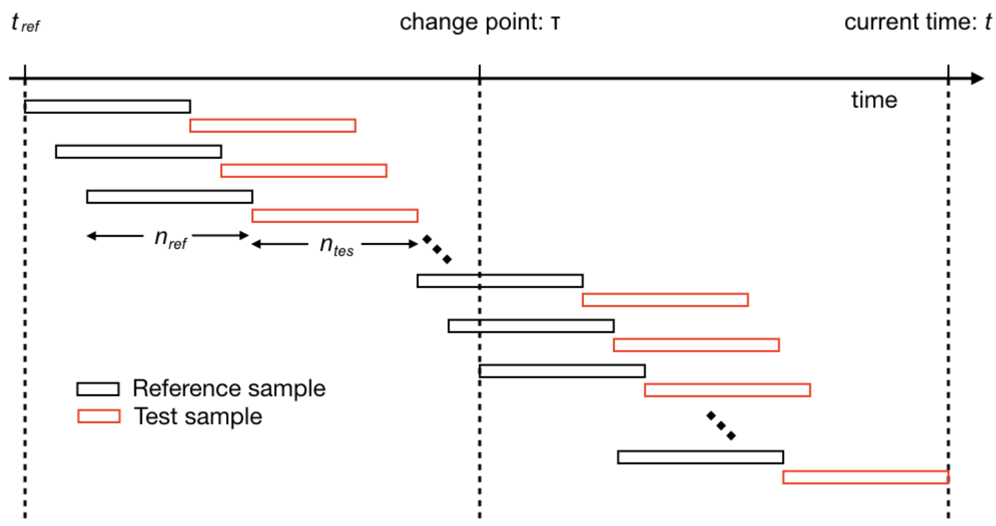


Figure 7: *The definition of the reference and test samples.*

This online sampling method allows us to compute $\hat{R}_t$ as time progresses. Certainly, there exits other ways of sampling, e.g. making n_{ref}, n_{tes} vary with time, or not forgetting the oldest observation whilst a new one is added at each time point. In what follows, we shall set n_{ref} and n_{tes} to be constant.

Now, we address the question of how to determine an appropriate threshold h. One way of doing so is to use the **Moving Average** (MA) and the **Moving Variance** (MV), defined as

$$MA_{ref}(s) := \frac{1}{n_{ref}} \sum_{i=t_{ref}(s)}^{t_{ref}(s)+n_{ref}-1} \hat{R}_i \quad , \quad MV_{ref}(s) := \frac{1}{n_{ref}-1} \sum_{i=t_{ref}(s)}^{t_{ref}(s)+n_{ref}-1} \left(\hat{R}_i - MA_{ref}(s)\right)^2 ,$$

$$MA_{tes}(s) := \frac{1}{n_{tes}} \sum_{i=t_{tes}(s)}^{t_{tes}(s)+n_{tes}-1} \hat{R}_i \quad , \quad MV_{tes}(s) := \frac{1}{n_{tes}-1} \sum_{i=t_{tes}(s)}^{t_{tes}(s)+n_{tes}-1} \left(\hat{R}_i - MA_{tes}(s)\right)^2 ,$$

where $t_{ref}(s)$, $t_{tes}(s)$ are the starting index of the reference and the test sample at time s, respectively. In other words, the MA and MV are the sample mean and sample variance of $\hat{R}_i$. The idea is to take notice when a certain number of consecutive $MA_{tes} \geq MA_{ref} + \sqrt{MV_{ref}}$ are observed. The intuition can be seen by plotting $\hat{R}_t$ against the time variable, t, as shown in **Figure 8**.

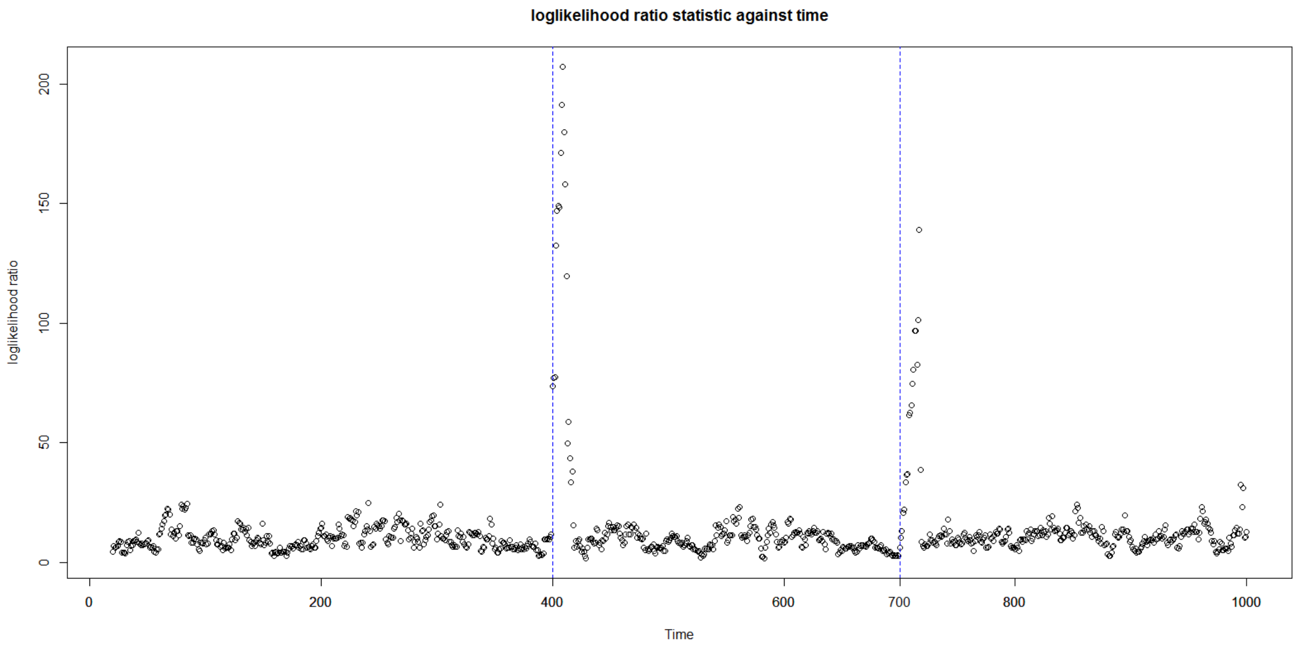


Figure 8: *The loglikelihood ratio vs time* ($N = 1000,\ p = 3$)*. Change point:* $\tau_1 = 400$, $\tau_2 = 700$.

A simulated dataset of size 1000 is simulated in generating this plot. Here, the design matrix consists a column of ones, and the rest covariates follow $Uniform(-10,\ 10)$. The error term chosen are normally distributed with mean $\underline{0}$ and covariance matrix $I^{(1000)}$. The true vector of parameters is 3-dimensional with the first two components being constant and the third being piecewise constant:

$$\underline{\beta}(t) = \begin{cases} (5.0, -8.0,\ 1.5)^T, & if\ 1 \leq t < 400\ ; \\ (5.0, -8.0,\ 0.4)^T, & if\ 400 \leq t < 700\ ; \\ (5.0, -8.0, -0.3)^T, & if\ 700 \leq t \leq 1000\ . \end{cases}$$

Hence, the true change points are $\tau_1 = 400$ and $\tau_2 = 700$, respectively. It is clear from the plot that $\hat{R}_t$ is close to zero with a small variance for most of the time, except near τ_1 and τ_2, where it experiences a sudden noticeable surge. It is therefore reasonable to pay attention when a sequence of 'abnormal' loglikelihood ratios is obtained, then the estimate of the change point is the index of the last element in the sequence subtracted by the length of the sequence, say m. The choice of m is still open. In this report, it is chosen to be n_{ref}, as this gives the most accurate empirical results. Note also that the region $MA_{tes} < MA_{ref} + \sqrt{MV_{ref}}$ form a band above the x-axis, and we are interested in the values $MA_{tes}(t)$ lying outside of it.

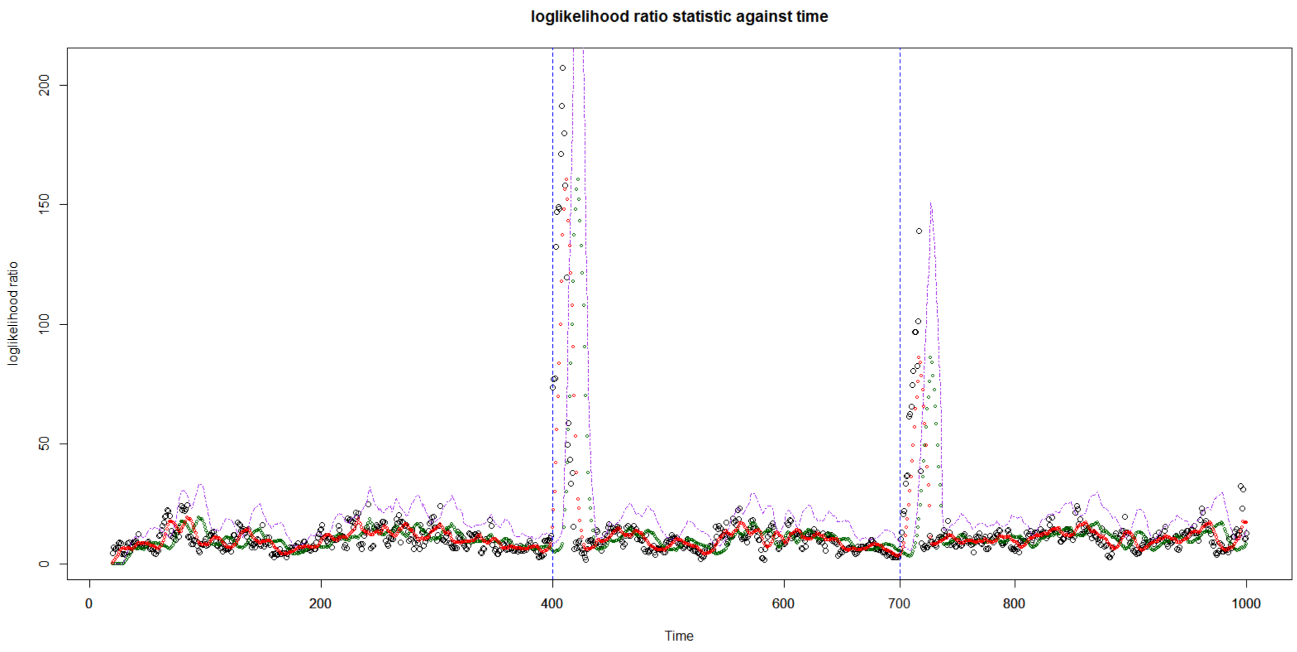


Figure 9: *The loglikelihood ratio vs time ($N = 1000,\ p = 3$). Dark green dots: the loglikelihood ratio computed from the reference sample. Red dots: the loglikelihood ratio computed from the test sample. Purple dotted line: the upper bound of the moving average band. Change point: $\tau_1 = 400$, $\tau_2 = 700$.*

Figure 9 shows the same plot as in **Figure 8** incorporated with the moving average band. It is clear that near the actual change points (which are unknown *a priori*), there is a sequence of MA_{tes} lying above the upper bound of the moving average band, thus suggesting a strong evidence of abrupt changes to the model. Here, the estimated change points are $\hat{\tau}_1 = 398$ and $\hat{\tau}_2 = 701$, which are very close to the real values.

The Quandt's MV band test has several advantages: i) the statistic can be computed in a recursive manner, ii) it is able to detect more than one change point for off-line data, and iii) it detects the time point of change with a remarkable accuracy on suitably choosing n_{ref} and n_{test}. However, as mentioned before, one major downside of this test is that the statistic has no asymptotic distribution, meaning that the best choice of some quantities (n_{ref}, n_{tes}, m etc.) differs from one case to another, and often has to be determined by trials.

4.2 The CUSUM

Recall that in **Section 3.2** we have defined the standardized residual associated with the i^{th} observation, r_i, to be the residual divided by its standard deviation, so that $r_i \sim N(0, \sigma^2)$ under the normal theory assumption. If there was a abrupt change in, say, the model mean, then $E[Y_i]$ would differ from $\underline{X}_i^T \underline{\beta}$, and r_i would therefore have non-zero expectation. On the other hand, if σ^2 was not always constant, then, although $E[r_i]$ is still zero, $Var(r_i)$ would be no longer constant. Brown *et al* [8] hence suggested a consistency-testing statistic, called the **CUSUM** statistic, which makes use of cumulative sums of the standardized residuals. In particular, suppose again the null hypothesis of consistent $\underline{\beta}$ or σ^2 before and after some time point τ is to be tested against the alternative hypothesis that there is actually a change. The **CUSUM** test statistic is defined as

$$W_t := \frac{1}{\hat{\sigma}} \sum_{i=p+1}^{t} r_i \ ,$$

where $\hat{\sigma} = \sqrt{RSS(t)/(t-p)}$ is the square root of the (weighted) LSE for σ^2 based on the first t observations.

It is obvious that $W_t \sim N(0, t-p)$. Therefore, it is suggested to reject H_0 for extreme values of W_t. Brown *et al* [9] further proposed a family of reference lines passing through the points $\{(p, a\sqrt{t-p})\ ,\ (N, 3a\sqrt{t-p})\}$, $\{(p, -a\sqrt{t-p})\ ,\ (N, -3a\sqrt{t-p})\}$, respectively, which take the form

$$y = \frac{2a}{\sqrt{t-p}}x + \frac{at-3ap}{\sqrt{t-p}} \quad , \quad y = -\frac{2a}{\sqrt{t-p}}x + \frac{3ap-at}{\sqrt{t-p}} \ ,$$

where a is the parameter depending on the significant level α. On fixing α, a can be found by solving the equation [8] (Some commonly used values can be found in **Table 4**):

$$Q(3a) + \exp(-4a^2)(1 - Q(a)) = \frac{1}{2}\alpha \ , \text{ where}$$
$$Q(z) = \frac{1}{\sqrt{2\pi}} \int_z^{\infty} \exp(-\frac{1}{2}u^2)du \ . \tag{4.2.1}$$

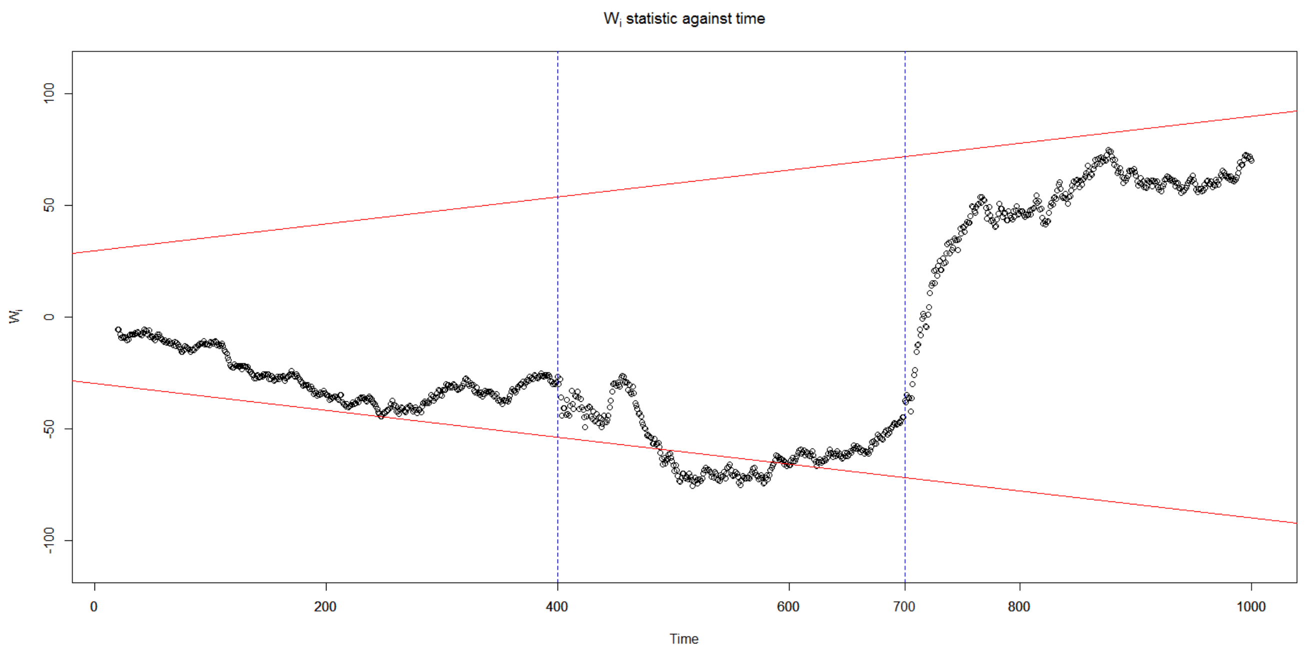


Figure 10: *The CUSUM statistic vs time ($N = 1000$, $p = 3$). Red lines: the reference lines given by* ***(4.2.1)*** *at significance level 5%. Change points: $\tau_1 = 400$, $\tau_2 = 700$.*

Figure 10 illustrates the CUSUM statistic against time computed from the same dataset as in **Figure 9**. Clearly, the W_t values deviate significantly from zero and exceed the reference lines near the change points, suggesting the existence of abrupt changes. However, despite that performing the CUSUM test only requires a little computational effort, it is hard to develop a strategy that uses the CUSUM statistic to concurrently estimate the change points τ_1 and τ_2, and this test seems to be very insensitive to small changes in the model, as discussed in **Section 4.4**.

α	a	a_1	a_2	a_3
10%	0.8499	1.07298	-0.66989	-0.58165
5%	0.9479	1.22387	-0.67001	-0.73517
2.5%	1.0365	1.35810	-0.67012	-0.88569
1%	1.1430	1.51743	-0.67027	-1.08477
0.5%	1.2172	1.62762	-0.67037	-1.23659

Table 11: *Critical values of the one-sided CUSUM and CUSUMSQ Test.*

4.3 The CUSUMSQ

The CUSUMSQ test [8], again proposed by Brown *et al*, uses the quantities

$$S_k = \frac{\sum_{i=p+1}^{k} r_i^2}{\sum_{i=p+1}^{t} r_i^2} \quad , \quad k = p+1, \; \dots \; , t,$$

and the test statistic is

$$\zeta_t = \max_{k=p+1,\dots,t} \left| S_k - \frac{k}{t-p} \right| \; ,$$

where t is again the current time. Under H_0, S_k can be shown to follow a beta distribution with mean $(k-p)/(t-p)$ [8]. This suggests that evidence is given against the null hypothesis for $\zeta_t > h$, for some threshold h. The computation of h given a significance level α is not trivial whatsoever, and Durbin managed to produce a table of significance values for small sample sizes in 1969 [10]. It is then extended by Edgerton and Wells [11], who gave a explicit formula to approximate h with a remarkable precision. In particular, the formula has the form

$$h = \frac{a_1}{\sqrt{t'}} + \frac{a_2}{t'} + \frac{a_3}{(t')^{3/2}} \; , \tag{4.3.1}$$

where $t' = (t-p)/2 - 1$ and a_1, a_2, a_3 are the parameters depending on α. If t' is not an integer it is suggested to interpolate linearly between the values $(t-p)/2 - 3/2$ and $(t-p)/2 - 1/2$ [8]. Some typical values are given in **Table 11**.

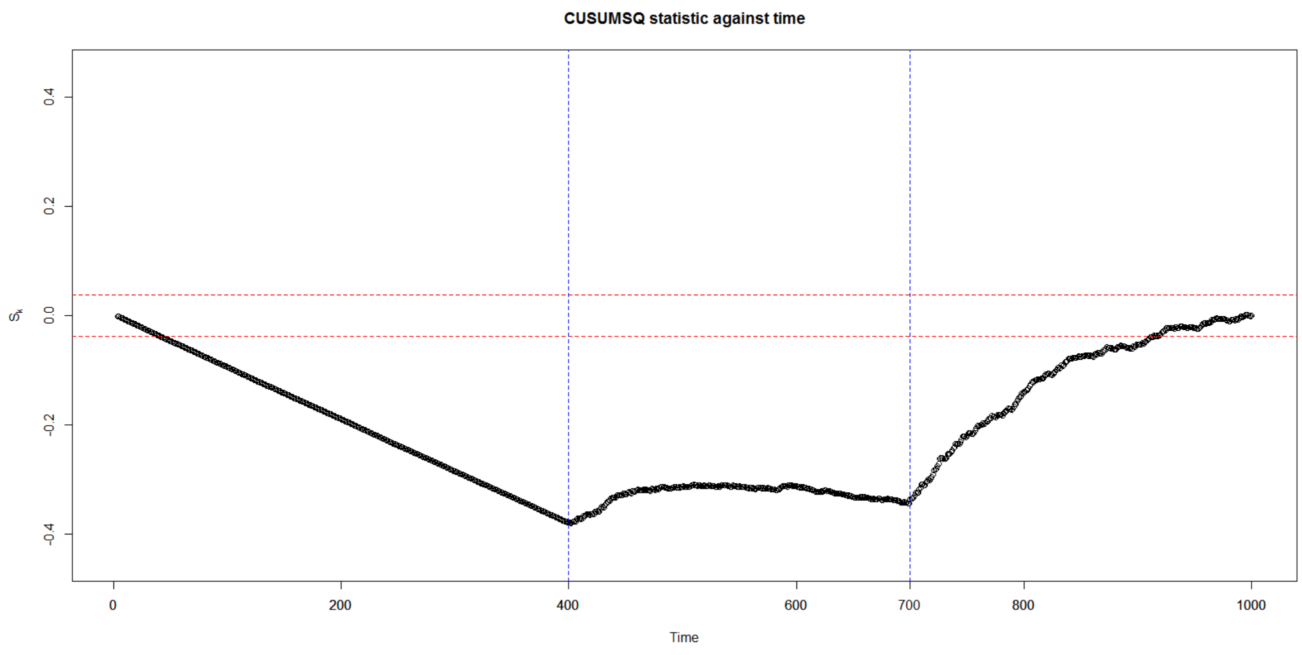


Figure 12: *The CUSUMSQ statistic vs time ($N = 1000$, $p = 3$). Red lines: the reference lines given by **(4.3.1)** at significance level 5%. Change points: $\tau_1 = 400$, $\tau_2 = 700$.*

The CUSUMSQ statistic S_k computed from the dataset used in **Figure 9** is plotted against time in **Figure 12**. One can see that S_t is well off the reference lines (marked as red) for most of the time points. Similar to the CUSUM, although this test successfully detects the existence of change points, the time points at which S_t crosses the reference line would not be good estimates of τ_1 and τ_2. It is tempting to use the 'local extrema' (roughly at $t = 400$ and 700 in **Figure 12**) as the estimates; however, it is only exploiting off-line data when we can identify this behaviour – for online data it is impossible to determine whether the current value is the most extreme one or not.

One natural approach of tackling this problem is to apply the loglikelihood test together with the CUSUMSQ test: the former is performed each time a new datum is fed; if it flags a potential change at, say, T, the latter is then implemented on the data upto T to double check the validity. The estimate $\hat{\tau}$ is chosen to be the time point at which the extremum of the CUSUMSQ statistic occurs. This method is proven empirically to work better than applying either of the two methods alone in general, as demonstrated in **Section 4.4**.

4.4 Performance Analysis

To compare the performance of the various methods that have been introduced so far, these tests are performed on a simulated dataset, where 600 samples of size $n = 300$ following $N(0, 1)$ are generated. Here, $p = 3$ and the covariates are drawn from *Uniform*(-10, 10). Half of the samples has one change point at $\tau = 150$, whilst the other half has a model consistent with time. The results are shown in **Table 13**.

Method	TP	FN	FP	TN	sen (%)	acc (%)	pre (%)	WMSE
Quandt's	300	0	19	281	100.00	96.83	94.04	274.21
CUSUM	17	283	11	289	5.67	51.00	60.71	20838.61
CUSUMSQ	300	0	64	236	100.00	89.33	82.42	156.90
Combined	290	10	7	293	96.70	97.17	97.64	1020.61

Table 13: *Performance analysis of various change-point detection methods. The Combined method uses a technique as described in* ***Section 4.3****. Sen: sensitivity. Acc: accuracy. Pre: precision.*

Several performance metrics are employed and reported in the table, which are defined as follows [12]:

Definition 4.1. *The sensitivity, also known as Recall or True Positive Rate, is defined as the portion of the change points that are correctly detected.*

$$sensitivity = \frac{TP}{TP + FN} \ .$$

Definition 4.2. *The accuracy is calculated as the ratio of the true classifications to all the data points.*

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \ .$$

Definition 4.3. *The precision is computed as the ratio of the true positive data points to the total points classified as change points.*

$$sensitivity = \frac{TP}{TP + FP} \ .$$

In addition, the column of *WSSE* represents the *weighted sum of squared error*, whose formula is given as

$$WSSE_j = \frac{1}{\sum_{k=1}^{300}(1 + I\{\hat{\tau}_{kj} < \tau\})} \sum_{k=1}^{300}(1 + I\{\hat{\tau}_{kj} < \tau\}) |\hat{\tau}_{kj} - \tau|^2 \ ,$$

with the intuition of the weights being that, in practice, one would prefer to predict a change point before the actual change occurs than after it. Here, $I\{\cdot\}$ is the usual indicator function and $j = 1, ..., 4$ stands for the four methods, respectively.

From the table, one can see that the Combined method is a rather conservative estimation. While being unable to detect any change point in 10 of the samples, it greatly reduces the number of false positives to 7, compared with 19 for the Quandt's and 764 for the CUSUMSQ.

Note that the WMSE of the CUSUMSQ is remarkably lower than the others. This suggests that, despite being very likely to produce false positives, the CUSUMSQ is able to estimate very accurately any existing change point. Furthermore, looking at the WMSE values alone, the Combined method seems to work poorly in terms of estimating the exact point of change. However, by investigating the estimates, one can conclude that this is due to the 10 change points that it missed, which are set to 0 in the calculation. Removing them make the WMSE become 279.94, which is very close to (although still higher than) the Quandt's and the CUSUMSQ.

Finally, the CUSUM method produces the least desirable results. It does not manage to detect most of the change points, despite giving only a few false positives. This may be due to its low sensitivity, and this method thus might be the least favourable one.

5 Chromosome Data

In this section, we implement the four change-point detection methods stated in **Section 4** to a real dataset *Chrom_13_GBM31* (available from the R package *changepoint*). This dataset is an excerpt of chromosome 13 in GBM31, with the latter being a particularly malignant type of brain tumor. It is taken from [13], in which the author has also studied it with other algorithms. *Chrom_13_GBM31* contains 797 pieces of data, each has 5 attributes (*GBM31, POS.start, etc*), and we are interested in how *GBM31* varies with time, which is assumed to follow a normal distribution with unknown mean and variance. The hypotheses are:

H_0 : *There are at least one abrupt change in the model, against*

H_1 : *The model is consistent.*

We apply the RLS algorithm to the data, where the forgetting factor λ is chosen to be 1. Furthermore, since the covariates do not assume significant dependence, the regularization parameter δ is set to be close to zero (0.0001 in this case). In other words, we are fitting an ordinary linear model. In principle, one would attempt to fit the model with with all available regressors (i.e., *Spot, POS.start and POS.end*). However, doing so in this case would result in an undesirable coefficient of determination (0.0656). By trial and error, the following model produces the most reasonable R^2 statistic (0.0739):

$$\textit{GBM31} = \beta_0 + \beta_1 \times \textit{POS.start} + \epsilon, \quad \epsilon \sim N(0, \sigma^2) \ . \tag{6.1.1}$$

Note that this R^2 value is still very close to zero. Nevertheless, given the assumptions of Gaussian settings and *iid.* observations, this is the best model we can find. Future studies could concentrate on fitting generalized recursive least-square models to free these constraints.

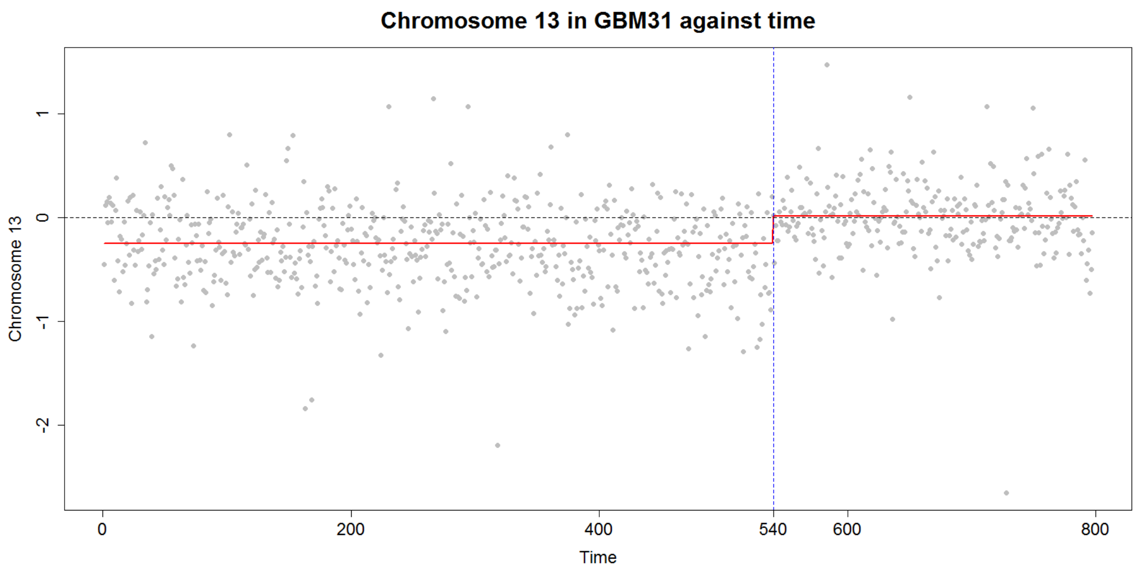


Figure 14: *Chromosome 13 in GBM31 over time. An abrupt change seems to have occurred at* $t \approx 540$.

The above plot shows the chromosome 13 data (*GBM31*) against time. One could see that it suggests an abrupt change might have occurred at time around 540. Additionally, there seems to be a few isolated departures, e.g. at $t = 163$ and 710. These outliers can be identified by finding those observations whose studentized residuals are large in absolute value, as stated in **Section 3**.

Implementing the four change-point detection methods yields the results as shown in Figure 15. As can be seen from the plot, almost all methods managed to capture the shift in nearly identical regions. *Quandt's*, however, identified an additional change point at $\hat{\tau} = 159$. This is likely because of the numerous individual outliers scattered around this time point. Indeed, by computing the studentized residuals, two nearby observations (at $t = 163$ and $t = 168$) have studentized residuals -4.138 and -3.946737, respectively, which are larger than the typical threshold, 3. Additionally, one can see that this issue is eliminated by *Combined* via further implementing the *CUSUMSQ* test.

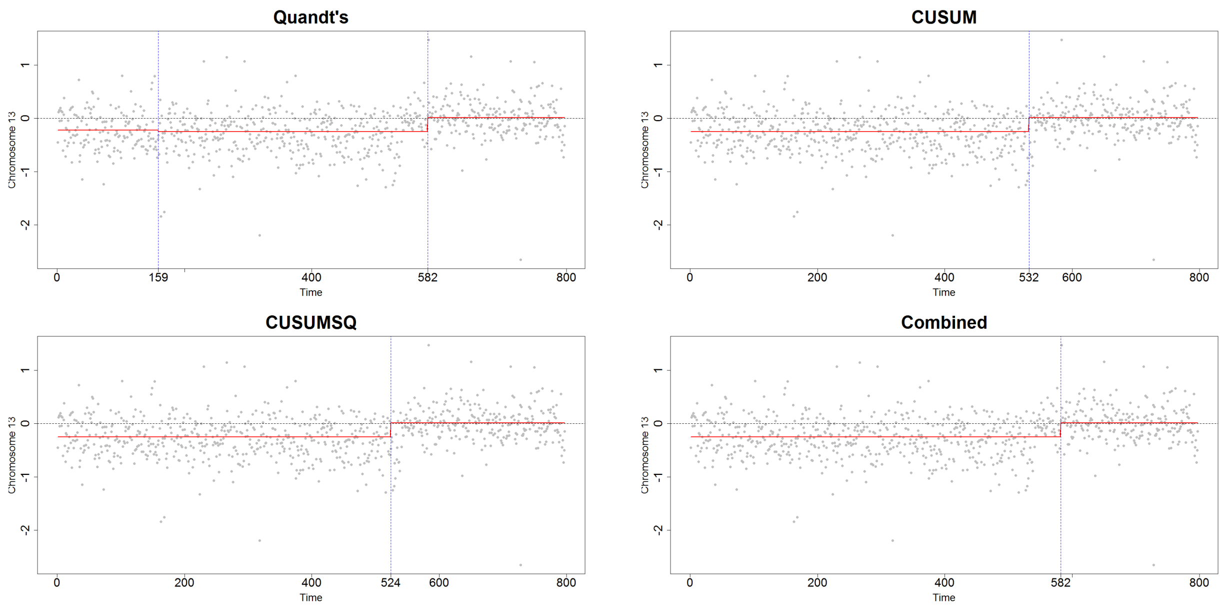


Figure 15: *Test results of the four change-point detection methods. Blue dotted line: estimated change points.*

Figure 16 illustrates the corresponding test statistics versus time. Note that once a change point is identified, the test statistics should be reset to zero and the algorithm should then restart, otherwise the result is likely to be greatly affected by previously computed values, as can be seen from the middle plot.

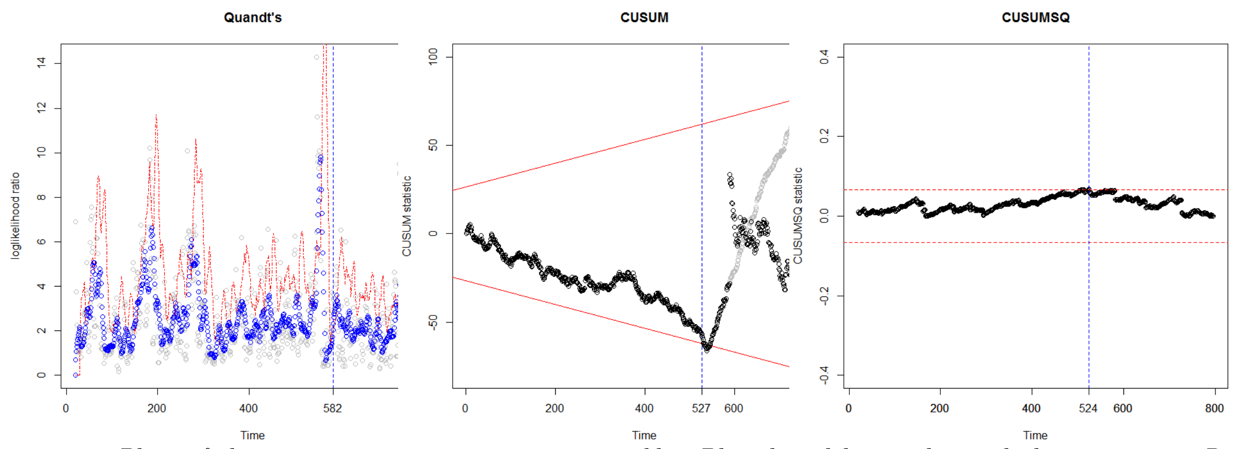


Figure 16: *Plots of the test statistics against time variable. Blue dotted lines: detected change points. Red (broken) lines: threshold values.* ***Left****: blue dots –* MA_{tes}*;* ***Middle****: black dots: CUSUM statistic,* W_t*, computed by restarting the algorithm after detection; gray dots:* W_t *computed by not restarting.* ***Right****: black dots: CUSUMSQ statistic* S_t*.*

6 Conclusion

In this report, we have presented the recursive least-square algorithm, some measures for outlier identifications and multiple change-point detection methods.

The recursive least-square algorithm can be remarkably useful when considering time-series data. A number of important goodness-of-fit measures, such as the weighted RSS, can be computed in an iterative manner, thereby saving computational effort. By suitably tuning the forgetting factor, this algorithm can also be implemented on datasets of an arbitrarily large size.

To identify potential outliers, one can compute recursively the leverages and the studentized residuals, then compare them with the typical thresholds mentioned in **Section 3**.

Finally, when applying with the RLS to detect potential change points, the Quandt's-with-MA-band and the Combined methods give the most reliable results amongst the four. Considering the pay-off between the computational effort and the precision of the estimates, the latter should be preferred in practice. However, one drawback shared by all the four algorithms is that they cannot distinguish change points from outliers. This issue can often be avoided by combining two or more tests, as demonstrated in the chromosome example.

7 Appendix.

7.1 Main Body.

```
rls<-function(response = NA, data = NA, lambda = NA, delta = NA, intercept = FALSE,
             cp = FALSE, n.tes = 10, n.ref = 10, alpha = 2){
  # Input: a dataframe with the top row being the most recent data set
  #                   with one column of data and the most recent one at the end
  # Output: the least square vector of predictors


  # Set no. of predictors
  data <- matrix(cbind(response, data), ncol = ncol(data)+1)
  p <- ncol(data)-1
  # Set no. of iterations
  n <- nrow(data)
  # Initialize a matrix for storing the LSE
  beta_hat.mat <- matrix(rep(NA, p*n), nrow = p)
  # Initialize a row vector for storing the RSS
  RSS.vec <- matrix(rep(NA, n), nrow = 1)
  # Initialize a row vector for storing R^2
  R_squared.vec <- matrix(rep(NA, n), nrow = 1)
  # Initialize a column vector for storing the estimated y(i) values
  y_hat.vec <- matrix(rep(NA, n), ncol = 1)

  # Initialize a matrix for storing the leverages
  # Column i of leverage.mat = leverages at time i
  leverage.mat <- matrix(rep(NA, n*n), ncol = n)
  # Initialize a matrix to locate any outliers
  outlier.mat <- matrix(rep(NA, n*n), ncol = n)
  # Initialize a matrix to store the studentized residuals
  stud_res.mat <- matrix(rep(NA, n*n), ncol = n)
  # Initialize a matrix to store the Cook's distance
  cook.mat <- matrix(rep(NA, n*n), ncol = n)

  # Initialize quantities for change-pt detection
  # Quandt's method
  count <- 0
  count.cts <- 0
  log.ratio <- matrix(rep(0,n),ncol = 1)
  ratio.bar <- matrix(rep(0,2*N),ncol=2)
  ratio.var <- matrix(rep(0,2*N),ncol=2)
  tao.hat <- numeric()
  # CUSUMSQ method
  cusumsq.vec <- matrix(rep(0,n),ncol = 1)
  cusumsq.max <- -Inf
  cusumsq.sum <- 0
  # Multiple models
  beta.vec.many <- numeric()


  ###############
  # Include the intercept term if specified
```

```
if (intercept == TRUE){
  data['Intercept'] <- rep(1,n)
  #Find the intercept column
  l = which(colnames(data) == 'Intercept')
  #Rearrange data: response, intercept, ...
  data <- data[, c(1, l, c(1:ncol(data))[c(-1,-l)])]
  p <- ncol(data)-1
}

###############
#Initialize the 0th iteration
beta_hat.vec <- as.matrix(rep(0,p), ncol=1)
D <- delta^(-1)*diag(p)

###############
# Initialize quantities for computing R^2
RSS <- 0
y.var <- 0
y.bar <- 0
###############

# Begin iterations
for (i in 1:n ){

  ###############
  #Extract the response vector at time i
  y <- response[i]
  #Extract the data vector at time i
  x.vec <- data[i,-1]
  ###############
  # Detect hange-point by default
  # if (i >= (n.ref+n.tes) && cp == TRUE){
  #   cp.summary <- rls.cp(i, switch, count n.ref, n.tes, log.ratio,
  #                         response, data[,-1], lambda, delta)
  #   switch <- cp.summary$switch
  #   log.ratio[i] <- cp.summary$log.ratio[i]
  #   if (switch == 1){
  #     print(c('Change point detected: i = ', i))
  #     #Restart initialization
  #     beta_hat.vec <- as.matrix(rep(0,p), ncol=1)
  #     D <- delta^(-1)*diag(p)
  #     # # Start from the current iteration
  #     # results <- rls.iterations(x.vec, beta_hat.vec, D, y, lambda)
  #   }
  # }
  if (i >= (n.ref+n.tes) && cp == TRUE){
    cp.summary <- rls.cp(i, switch, count,count.cts, n.ref, n.tes, log.ratio,
        ratio.bar,
                         ratio.var, response, data[,-1], lambda,
                        delta,beta_hat.vec, D, tao.hat, cusumsq.vec, cusumsq.sum,
                            alpha)
    # Set the next iteration
    count <- cp.summary$count
    count.cts <- cp.summary$count.cts
    log.ratio[i] <- cp.summary$log.ratio[i]
    ratio.bar[i,] <- cp.summary$ratio.bar[i,]
    ratio.var[i,] <- cp.summary$ratio.var[i,]
    beta_hat.vec <- cp.summary$beta_hat.vec
    D <- cp.summary$D
    tao.hat <- cp.summary$tao.hat

    beta_hat.vec.beforecp <- cp.summary$beta_hat.vec.beforecp
    if (length(beta_hat.vec.beforecp)>0){
      beta.vec.many <- cbind(beta.vec.many,beta_hat.vec.beforecp)
    }
  }
  ###############
  #Perform the current iteration
  results <- rls.iterations(x.vec, beta_hat.vec, D, y, lambda)
  ###############
  #Set the next iteration
  beta_hat.vec <- results$beta.hat
  D <- results$D
  ###############
  # Compute the leverage of the observations
  for (j in 1:i){
    leverage.mat[j,i] <- data[j,-1] %*% D %*% data[j,-1]
  }
  # Compute the studentized residuals
  e.vec <- (response[1:i] - data[1:i,-1]%*%beta_hat.vec)
  weighted.e.vec <- e.vec * sqrt(c(lambda^((i-1):0)))
  sigma.hat <- as.numeric(t(weighted.e.vec)%*%weighted.e.vec)/(n-p)
  leverage.vec <- leverage.mat[1:i,i]
  stud_res.vec <- e.vec/sqrt(abs(1-leverage.vec)*sigma.hat)
  stud_res.mat[1:i,i] <- stud_res.vec
  # Compute the Cook's distance
  cook.mat[1:i,i] <- stud_res.vec^2 * leverage.vec/((1-leverage.vec)*p)

  # # Only store the latest leverage
  # y.hat <- results$y_hat.val
  # leverage <- x.vec %*% D %*% x.vec
  # leverage.mat[i,N] <- leverage
  # e.vec <- response[1:i] - data[1:i,-1]%*%beta_hat.vec
  # sigma.hat <- as.numeric(t(e.vec)%*%(e.vec))/(i-p)
  # stand_res.vec <- e.vec/sqrt(1-leverage)
  # stud_res.mat[i,N] <- stand_res.vec/sqrt(sigma.hat)
  # cook.mat[i,N] <- stand_res.vec^2 * leverage/((1-leverage)*p)
  ###############
  # Compute R^2
  # y.bar <- y.bar + 1/i * (y - y.bar)
  # RSS <- lambda*RSS + 1/i * ((y - x.vec %*% beta_hat.vec)^2 - lambda*RSS)
  # y.var <- lambda*y.var + 1/i * ((y - y.bar)^2 - lambda*y.var)
  # R_squared <- (y.var - RSS)/y.var

  # Compute R^2
  y.bar <- y.bar + 1/i * (y - y.bar)
```

```
    zeta <- results$zeta
    RSS <- lambda*RSS + zeta*(y - x.vec %*% beta_hat.vec)
    if (i == 1){
      RSS <- (y - x.vec %*% beta_hat.vec)^2
    }
    y.var <- t(response[1:i] - y.bar)%*%(response[1:i] - y.bar)
    R_squared <- (y.var - RSS)/y.var
    ###############
    # Store statistic for the CUSUMSQ
    zeta <- results$zeta
    cusumsq.i <- (zeta/sqrt(1+leverage.vec[i]))^2
    cusumsq.vec[i] <- cusumsq.i
    cusumsq.sum <- cusumsq.sum + cusumsq.i
    ###############
    # Store the results into corresponding matrices
    beta_hat.mat[,i] <- beta_hat.vec
    RSS.vec[,i] <- RSS
    y_hat.vec[i,] <- results$y_hat.val
    R_squared.vec[,i] <- R_squared
  }

  ###############
  # Flag those with a high leverage
  # Remove the empty entries in outlier.mat
  outlier.mat <- which(leverage.mat > 2*p/n, arr.ind = TRUE)
  outlier.mat <- which(leverage.mat[,n] > 2*p/n, arr.ind = TRUE)

  # Return the results
    # Cp detection results
  cp.summary <- list('log.ratio' = log.ratio,
                     'ratio.bar' = ratio.bar,
                     'ratio.var' = ratio.var,
                     'tao.hat' = tao.hat
                     )
  gof.summary <- list('RSS.vec' = RSS.vec,
                      'R2.vec' = R_squared.vec,
                      'leverage.mat' = leverage.mat,
                       'outlier.mat' = outlier.mat,
                       'Studentized.residuals' = stud_res.mat,
                       'Cook.distance' = cook.mat
                     )

  #############
  beta.vec.many <- cbind(beta.vec.many,beta_hat.vec)
  #############
  rls.summary <- list('beta_hat.vec' = as.matrix(beta_hat.mat[,n],ncol = 1),
                       'beta_hat.mat' = beta_hat.mat,
                       'y_hat.vec' = y_hat.vec,
                       'RSS' = RSS.vec[,n],
                       'RSS.vec' = RSS.vec,
                       'R2.vec' = R_squared.vec,
                       'leverage.vec' = leverage.mat[,n],
                       'leverage.mat' = leverage.mat,
                       'outlier.mat' = outlier.mat,
                       'Studentized.residuals' = stud_res.mat,
                       'Cook.distance' = cook.mat,
                       'gof.summary' = gof.summary,
                       'cp.summary' = cp.summary,
                       'beta.vec.many' = beta.vec.many
                       )


  return(rls.summary)
}
```

7.2 Iteration Step.

```
rls.iterations<-function( x.vec = NA, beta_hat.vec = NA, D = NA,
                          y.val = NA, lambda = NA){
  # Input: n = the nth iteration, x.vec = the nth data vector x.vec(n)
  #       beta_hat.vec = beta.hat.vec(n-1), y.val = data vector at time n, D =
     D(n-1)

  gamma.vec <- lambda^(-1) * D %*% x.vec/as.numeric((1 + lambda^(-1)*t(x.vec) %*% D
     %*% x.vec))
  zeta <- as.numeric(y.val - t(beta_hat.vec) %*% x.vec)
  beta_hat.vec <- beta_hat.vec + gamma.vec * zeta
  D <- lambda^(-1)*D - lambda^(-1) * gamma.vec %*% t(x.vec) %*% D

  # Compute estimated Y value
  y_hat.val <- as.numeric(t(beta_hat.vec) %*% x.vec)

  output <- list('beta.hat' = beta_hat.vec,
              'D' = D,
              'y_hat.val' = y_hat.val,
              'zeta' = zeta
              )
  return(output)
}
```

7.3 Change-point Detection Using the Quandt's Method.

```
rls.cp <- function(i=NA, switch = NA, count=NA, count.cts=NA, n.ref = NA, n.tes =
    NA, log.ratio = NA, ratio.bar = NA,
                 ratio.var = NA, y=NA, data=NA, lambda=NA,
                 delta=NA,beta_hat.vec=NA, D=NA, tao.hat=NA, cusumsq.vec=NA,
                    cusumsq.sum=NA, alpha=NA){
  # Input: h = open reference point (h>0)
  # Output:

  # switch = 2 if a cp was detected previously, 1 if a cp was detected at this
     iteration, 0 if not.
  switch.new <- 0
```

```
p <- ncol(data)
beta_hat.vec.beforecp <- numeric()

rls.fit.ref <- rls.fast(y[(i-n.tes-n.ref+1):(i-n.tes)],
   data[(i-n.tes-n.ref+1):(i-n.tes),], lambda, delta)
rss.ref <- rls.fit.ref$RSS

# beta.hat.ref <- rls.fit.ref$beta_hat.vec
# res.ref <- y[(t-n.tes+1):t]-data[(t-n.tes+1):t,] %*% beta.hat.ref
# rss.ref <- t(res.ref)%*%res.ref

rls.fit.tes <- rls.fast(y[(i-n.tes+1):i], data[(i-n.tes+1):i,], lambda, delta)
rss.tes <- rls.fit.tes$RSS

sigma.sq.hat <- as.numeric(rss.ref+rss.tes)/(n.tes+n.ref-p)
log.ratio[i] <- ((n.ref-n.tes)/2)*log(2*pi) + ((n.ref-n.tes)/2)*log(sigma.sq.hat)
   + (1/2)*rss.tes + (1/2)*rss.ref

###########
# Determine the threshold using MA
sample.size <- min(i-n.ref-n.tes+1,n.ref+n.tes )
# Ref MA
ratio.bar[i,1] <- mean(log.ratio[(i-n.tes-n.ref+1):(i-n.tes)])
ratio.var[i,1] <- (
   1/max((sample.size-1),1)*(sum(log.ratio[(i-n.tes-n.ref+1):(i-n.tes)]^2) +
                                       sample.size*ratio.bar[i,1]^2 -
                                       2*ratio.bar[i,1]*sum(log.ratio[(i-n.tes-n.ref+1):
                                          )
# Tes MA
ratio.bar[i,2] <- mean(log.ratio[(i-n.tes+1):i])
# + sample.size*ratio.bar[i,2]^2 - 2*ratio.bar[i,2]*sum(log.ratio[(i-n.tes+1):i])

# Compute the threshold value
ratio.ref.ub <- ratio.bar[i,1]+sqrt(ratio.var[i,1])
if (i >= (n.ref*3+n.tes) && ratio.bar[i,2] > ratio.ref.ub){
  # Start counting
  count <- count + 1
}
else {
  count <- 0
}

#######
# Raise an alarm if a change point is confirmed.
if (count == (n.tes )){

  # Double check with the CUSUMSQ test
  cusumsq.results <- rls.cusumsq(p, i, cusumsq.vec, cusumsq.sum, alpha)
  cusumsq.switch <- as.numeric(cusumsq.results$switch)
  cusumsq.stat <- cusumsq.results$cusumsq.stat
  critical.value <- cusumsq.results$critical.value
  significance.level <- cusumsq.results$significance.level
```

```
    # Confirm the change point if it passes the two tests
    if (cusumsq.switch == 1){
      cp <- (i-n.tes+1)
      # print(paste('Change point detected: i = ', cp))
      # print(paste('CUSUMSQ statistic: ', format(cusumsq.stat, digits = 6),
      #           '(Critical value: )', format(critical.value, digits = 6),
      #           'Rejected at significance level: ', significance.level))

      tao.hat <- cbind(tao.hat,cp)
      # Reset count for the next cp
      count <- 0
      # Reset initialization
      beta_hat.vec.beforecp <- beta_hat.vec
      beta_hat.vec <- as.matrix(rep(0,p), ncol=1)
      D <- delta^(-1)*diag(p)
    }
  }

  ########
  # Detect continuous cp
  if (i >= (n.ref*2+n.tes) && ratio.bar[i,2] >= ratio.bar[(i-1),2]){
    count.cts <- count.cts + 1
  }
  else{
    count.cts <- 0
  }

  if (count.cts == n.tes){
    # print(c('Continuous change point detected: i = ', i))
    count.cts <- 0
  }

  cp.summary <- list('switch' = switch.new,
                     'count' = count,
                     'count.cts' = count.cts,
                     'log.ratio' = log.ratio,
                     'ratio.bar' = ratio.bar,
                     'ratio.var' = ratio.var,
                     'ratio.ref.ub'= ratio.ref.ub,
                     'beta_hat.vec' = beta_hat.vec,
                     'D' = D,
                     'tao.hat' = tao.hat,
                     'beta_hat.vec.beforecp' = beta_hat.vec.beforecp
                     )

  return(cp.summary)

}
```

7.4 Change-point Detection Using the CUSUMSQ Method.

```
rls.cusumsq <- function(p=NA, i=NA, cusumsq.vec=NA, cusumsq.sum=NA, alpha=NA){
```

```
# Input:
# Output: The CUSUMSQ statistic and decision rule.

# Write up for multiple p-values
if (alpha == 1){
  a1 <- 1.0729830
  a2 <- -0.6698868
  a3 <- -0.5816458
  significance.level <- '10%'
}
else if (alpha == 2){
  a1 <- 1.2238734
  a2 <- -0.6700069
  a3 <- -0.7351697
significance.level <- '5%'
}
else if (alpha == 3){
  a1 <- 1.3581015
  a2 <- -0.6701218
  a3 <- -0.8858694
  significance.level <- '2.5%'
}
else if (alpha == 4){
  a1 <- 1.5174271
  a2 <- -0.6702672
  a3 <- -1.0847745
  significance.level <- '1%'
}
else {
  a1 <- 1.6276236
  a2 <- -0.6703724
  a3 <- -1.2365861
  significance.level <- '0.5%'
}

# 0 = H_0 not rejected, 1 = H_0 rejected
switch <- 0

# Compute the critical value
c <- a1/sqrt(i) + a2/i + a3/i^{3/2}

mean <- (c((p+1):i)-p)/(i-p)
cusumsq.stat <- max(abs(cumsum(cusumsq.vec[(p+1):i])/cusumsq.sum - mean))

if (cusumsq.stat > c){
  switch <- 1
}

summary <- list( 'switch' = switch,
                'cusumsq.stat' = cusumsq.stat,
                'critical.value' = c,
                'significance.level' = significance.level
               )
```

```
 }
```

7.5 Change-point Detection Performance Analysis.

```
# CUSUM Performance Analysis
# time needed for k=600: 1h
start.time <- Sys.time()
tao.hat <- matrix(rep(0, 600*4), nrow=4)
for (k in 1:600){

  N <- 501
  p <- 3
  intercept <- matrix(rep(1,N), ncol = 1)
  data <- cbind(intercept, matrix(c(runif(n=N*(p-1), min=-10,max=10)),ncol=(p-1)))
  beta <- matrix(rep(NA, N*(p-1)),ncol = (p-1))
  beta <- t(cbind(rep(5,N),beta))
  y <- matrix(rep(NA,N),ncol = 1)
  # #########
  # beta[2,]<- -8
  # beta[3,]<-3
  beta[2,]<- -8
  beta[3,1:249]<-1.5
  beta[3,250:N]<- -0.3

  # if (k > 300){
  #   beta[3,] <- 1.5
  # }
  #########
  # beta[2,]<- -8
  # beta[3,1:399]<-1.5
  # beta[3,400:699]<-0.4
  # beta[3,700:1000]<--0.3
  #########
  for (i in 1:N){
    y[i] <- data[i,]%*%beta[,i]
  }
  y <- y + matrix(c(rnorm(n=N, mean=0, sd=1)), ncol=1)

  ###########
  # CUSUM
  lambda<- 1
  delta <- 0.00001
  res <- matrix(rep(0, N*N), ncol=N)
  cusum.stat <- matrix(rep(0, N), ncol=1)
  cusum.count <- 0
  a<- 0.9479
  ref.val <- matrix(rep(0,N),ncol=1)
  ref.val <- 2*a/sqrt(N-p) * (1:N) + (a*N-3*a*p)/sqrt(N-p)
  ###########
  # Quandts
  n.ref <- 20
  n.tes <- 20
```

```
log.ratio <- matrix(rep(0,N),ncol=1)
ratio.bar <- matrix(rep(0,2*N),ncol=2)
ratio.var <- matrix(rep(0,2*N),ncol=2)
quandts.count <- 0
quandts.stop <- 0
###########
# Quandts improved
quandts.imp.stud.res <- matrix(rep(0, N),ncol=1)
quandts.imp.count <- 0
###########
# CUSUMSQ
rls.fit.sq <- rls.fast(response = y, data = data, lambda = 1,delta = 0.00001)
beta.hat.mat <- rls.fit.sq$beta_hat.mat
stud.res <- matrix(rep(0, N),ncol=1)


# start.time <- Sys.time()
for (t in (n.ref+n.tes):N){
    #############
    #CUSUM
    rls.fit <- rls.fast(y[1:t], data[1:t,], lambda, delta)
    # res[1:t,t] <- matrix(diag(rls.fit$Studentized.residuals), ncol=1)
    # cusum.stat[t] <- sum(res[(p+1):t,t])
    cusum.res <- y[t] - (rls.fit$y_hat.vec)[t]
    # cusum.res <- y[t] - data[t,]%*%rls.fit$beta_hat.vec
    cusum.lev <- data[t,]%*%solve(t(data[1:t,]) %*% data[1:t,])%*%data[t,]
    res[1,t] <- cusum.res/sqrt(1-cusum.lev)
    cusum.sigma.sq.hat <- (rls.fit$RSS)/(t-p)
    # cusum.stat[t] <- sum(res[1,])/sqrt(cusum.sigma.sq.hat)
    cusum.stat[t] <- sum(res[1,])/sqrt(cusum.sigma.sq.hat)

    if (abs(cusum.stat[t]) > ref.val[t] && cusum.count == 0){
      tao.hat[1, k] <- t
      cusum.count <- 1
    }

    # ###########
    # Quandts
    rls.fit.ref <- rls.fast(y[(t-n.tes-n.ref+1):(t-n.tes)],
       data[(t-n.tes-n.ref+1):(t-n.tes),], lambda, delta)
    beta.hat.ref <- rls.fit.ref$beta_hat.vec
    res.ref <- y[(t-n.tes+1):t]-data[(t-n.tes+1):t,] %*% beta.hat.ref
    rss.ref <- t(res.ref)%*%res.ref

    rls.fit.tes <- rls.fast(y[(t-n.tes+1):t], data[(t-n.tes+1):t,], lambda, delta)
    rss.tes <- rls.fit.tes$RSS

    sigma.sq.hat <- as.numeric(rss.ref+rss.tes)/(n.tes+n.ref-p)
    log.ratio[t] <- ((n.ref-n.tes)/2)*log(2*pi) +
      ((n.ref-n.tes)/2)*log(sigma.sq.hat) + (1/2)*rss.tes + (1/2)*rss.ref

    sample.size <- min(t-n.ref-n.tes+1,n.ref+n.tes )
    # Ref
```

```
ratio.bar[t,1] <- mean(log.ratio[(t-n.tes-n.ref+1):(t-n.tes)])
ratio.var[t,1] <- (
    1/max((sample.size-1),1)*(sum(log.ratio[(t-n.tes-n.ref+1):(t-n.tes)]^2)
                                              + sample.size*ratio.bar[t,1]^2
                                              -
                                                 2*ratio.bar[t,1]*sum(log.ratio[(t-n.tes-n.re
                                                 )
# Tes
ratio.bar[t,2] <- mean(log.ratio[(t-n.tes+1):t])
ratio.var[t,2] <- ( 1/max((sample.size-1),1)*(sum(log.ratio[(t-n.tes+1):t]^2)
                                              + sample.size*ratio.bar[t,2]^2
                                              -
                                                 2*ratio.bar[t,2]*sum(log.ratio[(t-n.tes+1):t
                                                 )

ratio.ref.ub <- ratio.bar[t,1]+sqrt(ratio.var[t,1])
if (t >= (n.ref*2+n.tes) && ratio.bar[t,2] > ratio.ref.ub && quandts.stop ==
    0){
  # Start counting
  quandts.count <- quandts.count + 1
}
else {
  quandts.count <- 0
}

if (quandts.count == n.tes){
  quandts.count <- 0
  tao.hat[2,k] <- (t-n.tes+1)
  quandts.stop <- 1
  quandts.imp.count <- n.tes
}
# ###########
# Quandts improved
if (t >= (n.ref*2+n.tes) && ratio.bar[t,2] > ratio.ref.ub){
  # Start counting
  quandts.imp.count <- quandts.count + 1
}
else {
  quandts.imp.count <- 0
}

if (quandts.imp.count == n.tes){
  for (l in ((p+1):t)){
  quandts.imp.beta.hat <- beta.hat.mat[,(t-1)]
  quandts.imp.stud.res[t] <- (y[t]-data[t,]%*%quandts.imp.beta.hat)/sqrt((1 +
      data[t,]%*%solve(t(data[1:t,]) %*% data[1:t,])%*%data[t,]))
  }

  quandts.imp.stud.res.sq <- quandts.imp.stud.res^2
  quandts.imp.S <- sum((quandts.imp.stud.res.sq[(p+1):t]))
  quandts.imp.s.vec <- cumsum(quandts.imp.stud.res.sq)/quandts.imp.S
  quandts.imp.mean <- (c((p+1):t)-p)/(t-p)
  quandts.imp.stat <- abs(quandts.imp.s.vec[(p+1):t] - quandts.imp.mean)
```

```
      quandts.imp.stat.max <- max(quandts.imp.stat)
      quandts.imp.tao.hat <- as.numeric((which(quandts.imp.stat ==
         quandts.imp.stat.max))[1])
      t.prime <- 0.5*(t-p)-1
      quandts.imp.c <- 1.3581015/sqrt(t.prime) -0.6701218/(t.prime)
         -0.8858694/(t.prime)^{3/2}

      if (quandts.imp.stat.max > quandts.imp.c){
        quandts.imp.count <- 0
        tao.hat[4,k] <- as.numeric((which(quandts.imp.stat ==
           quandts.imp.stat.max))[1])
      }

      quandts.imp.count <- 0
    }

    ###########
    # CUSUMSQ
    beta.hat <- beta.hat.mat[,(t-1)]
    # cusumsq.res <- y[1:i]-data[1:i,]%*%beta.hat
    stud.res[t] <- (y[t]-data[t,]%*%beta.hat)/sqrt((1 +
       data[t,]%*%solve(t(data[1:t,]) %*% data[1:t,])%*%data[t,]))


    ###########
}
# print(Sys.time()-start.time)
# ###########
# CUSUM
# Keep commented out
# if (cusum.count == 0){
#   tao.hat[1, k] <- 0
# }
# Quandts
if (quandts.stop == 0){
  tao.hat[2, k] <- 0
}
# Quandts improved
# Keep commented out
# if (quandts.imp.count == 0){
#   tao.hat[4, k] <- 0
# }
# CUSUMSQ
stud.res.sq <- stud.res^2
S <- sum((stud.res.sq[(p+1):N]))
s.vec <- cumsum(stud.res.sq)/S
mean <- (c((p+1):N)-p)/(N-p)
cusumsq.stat <- abs(s.vec[(p+1):N] - mean)
cusumsq.stat.max <- max(cusumsq.stat)
cusumsq.tao.hat <- as.numeric((which(cusumsq.stat == cusumsq.stat.max))[1])
N.prime <- 0.5*(N-p)-1
c <- 1.3581015/sqrt(N.prime) -0.6701218/(N.prime) -0.8858694/(N.prime)^{3/2}
if (cusumsq.stat.max > c){
    tao.hat[3, k] <- cusumsq.tao.hat
  }
  if (cusumsq.stat.max < c || cusumsq.stat.max == c) {
    tao.hat[3, k] <- 0
  }
  ################
  # plot((n.ref+n.tes):N,cusum.stat[(n.ref+n.tes):N], ylim=c(-60,60))
  # abline((-3*a*p+a*N)/sqrt(N-p),2*a/sqrt(N-p))
  # abline((3*a*p-a*N)/sqrt(N-p),-2*a/sqrt(N-p))
  #
  # c <- 1.3581015/sqrt(248) -0.6701218/(248) -0.8858694/(248)^{3/2}
  # plot((n.ref+n.tes):N, cusumsq.stat[(n.ref+n.tes):N], ylim=c(-1,1))
  # abline(c,0)
  # abline(-c,0)
  # # what the hell are the thresholds??
  # # The original ones for cusum are correct
  # # The ones for cusumsq are incorrect
  print(tao.hat[,k])
}
```

References

[1] S.S. Haykin. *Adaptive Filter Theory*. Pearson Education, 2008. ISBN 9788131708699. URL https://books.google.co.uk/books?id=MdDi_PF7gMsC.

[2] Simon Haykin. In *Adaptive filter theory*, Always learning, chapter 9 & 10. Pearson Education, fifth edition, 2014. ISBN 027376408x.

[3] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*, volume 103 of *Springer Texts in Statistics*. Springer New York, New York, NY, 2013. ISBN 978-1-4614-7137-0.

[4] P. McCullagh and J.A. Nelder. *Generalized linear models*. Monographs on statistics and applied probability ; 37. Chapman and Hall, London, 2nd ed. edition, 1989. ISBN 0412317605.

[5] David C. Hoaglin and Roy E. Welsch. The Hat Matrix in Regression and ANOVA. *The American Statistician*, 32(1):17–22, 1978. doi: 10.1080/00031305.1978.10479237.

[6] Richard E. Quandt. The Estimation of the Parameters of a Linear Regression System Obeying Two Separate Regimes. *Journal of the American Statistical Association*, 53(284): 873–880, 1958. doi: 10.1080/01621459.1958.10501484.

[7] Richard E. Quandt. Tests of the Hypothesis That a Linear Regression System Obeys Two Separate Regimes. *Journal of the American Statistical Association*, 55(290):324–330, 1960. doi: 10.1080/01621459.1960.10482067.

[8] R. L. Brown, J. Durbin, and J. M. Evans. Techniques for Testing the Constancy of Regression Relationships over Time. *Journal of the Royal Statistical Society. Series B (Methodological)*, 37(2):149–192, 1975. ISSN 00359246. URL http://www.jstor.org/stable/2984889.

[9] Yoshinobu Kawahara and Masashi Sugiyama. *Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation*, pages 389–400. doi: 10.1137/1.9781611972795.34.

[10] J. Durbin. Testing for Serial Correlation in Least-Squares Regression When Some of the Regressors are Lagged Dependent Variables. *Econometrica*, 38(3):410–421, 1970. ISSN 00129682, 14680262.

[11] Edgerton David and Wells Curt. Critical values for the cusumsq statistic in medium and large sized samples. *Oxford Bulletin of Economics and Statistics*, 56(3):355–365. doi: 10.1111/j.1468-0084.1994.mp56003007.x.

[12] Samaneh Aminikhanghahi and Diane J. Cook. A survey of methods for time series change point detection. *Knowledge and Information Systems*, 51(2):339–367, May 2017. ISSN 0219-3116. doi: 10.1007/s10115-016-0987-z. URL https://doi.org/10.1007/s10115-016-0987-z.

[13] Weil R. Lai, Mark D. Johnson, Raju Kucherlapati, and Peter J. Park. Comparative analysis of algorithms for identifying amplifications and deletions in array CGH data. *Bioinformatics*, 21(19):3763–3770, 2005. URL http://dx.doi.org/10.1093/bioinformatics/bti611.

[14] Muggeo Vito M. R. Estimating regression models with unknown break-points. *Statistics in Medicine*, 22(19):3055–3071. doi: 10.1002/sim.1545.

[15] Johannes Hofrichter. *Change Point Detection in Generalized Linear Models.* PhD thesis, Graz University of Technology, January 2007.

[16] Bang Yong Sohn and Guk Boh Kim. Detection of outliers in weighted least squares regression. *Korean Journal of Computational & Applied Mathematics*, 4(2):441–452, Aug 1997. ISSN 1865-2085. doi: 10.1007/BF03014491.

[17] Johan Andersson. Locating Multiple Change-Points Using a Combination of Methods. Master's thesis, KTH Royal Institute of Technology, School of Engineering Sciences (SCI), Mathematics (Dept.), Sweden, 2014. URL `http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-146271`.

[18] Werner Ploberger and Walter Krmer;. The Local Power of the CUSUM and CUSUM of Squares Tests. *Econometric Theory*, 6(3):335347, 1990. doi: 10.1017/S0266466600005302.

[19] Samaneh Aminikhanghahi and Diane J. Cook. A survey of methods for time series change point detection. *Knowledge and Information Systems*, 51(2):339–367, May 2017. ISSN 0219-3116. doi: 10.1007/s10115-016-0987-z.