Statistical Analysis in Neuroimaging
Structural Breaks

Part III Essay

Contents

1 Introduction

There has recently been an explosion in research into neuroimagining techniques. These techniques generate a large amount of data to be analysed. There has also been a great deal of recent work in Big Data analysis; dealing with very large data sets. As a consequence of the growth in these areas, there has been much research into developing statistical methods which analyse data arising from neuroimaging.

One instance where statistical techniques need to be applied is structural breaks in fMRI data. FMRI stands for Functional Magnetic Resonance Imaging and is a neuroimaging procedure which measures activity in the brain by considering changes in blood flow. Structural breaks or changepoints refer to the locations of changes in a given time series. We are often interested in detecting when a change in the state of the brain occurs. FMRI gives us data which we can analyse to detect these changes.

The main form of fMRI is known as blood-oxygen-level dependent (BOLD) response developed by Ogawa et al (1990). This technique takes advantage of the fact that when neurons in the brain are active they need energy in the form of sugar and oxygen. The way that the brain responds to this is to channel oxygen to active neurons. This means that active regions of the brain contain larger amounts of oxygenated blood than inactive regions. The BOLD technique is able to detect this by taking advantage of the fact that oxygenated and de-oxygenated blood have different magnetic properties. It is thus able to generate a map of the active areas of the brain. In doing so the brain is divided into cubelike regions known as voxels and a measure of activity is returned for each voxel. Thus we obtain a multi-dimensional time series as we have univariate data over time for each voxel.

The activity occurring in voxels tends to be highly correlated. Functional connectivity refers to the dependence between the time series at each voxel. Given the data, this can be estimated by calculating a correlation matrix. Estimating the functional connectivity gives us an idea of the dependence between voxels. Taking the voxels to be nodes and the correlation matrix to be an adjacency matrix allows us to treat the brain as a multivariate time series network. In such a network an edge connecting a given pair of nodes represents the dependence between the times series associated with each node.

Networks result in a number of interesting change-point problems as one is often interested in identifying when a change in the network structure occurs. There are many real world networks, examples include the internet, neural networks, communication networks, social media networks, citation networks and many more. There is currently a great deal of research into changepoint detection and networks in separate contexts, but there is a lot of scope for using change-point detection methods in combination with network analysis to solve problems of interest.

One interesting example of identifying structural breaks in networks is developed in

Cribben and Yu (2015). This particular paper combines community detection methods with statistical techniques such as bootstrapping and change-point analysis to create the Network Changepoint Detection (NCPD) Algorithm. This algorithm detects changes in the community structure of a network. They apply this method to resting state fMRI data to illustrate how the algorithm can be used to identify changes in the brain. (Resting state fMRI data refers to fMRI data collected while the individual is in a resting state: not performing a specific task or solving a problem.)

In this essay we look at a number of areas to do with changepoint analysis and hypothesis testing. Section 2 introduces the concept of changepoints and the Binary Segmentation algorithm as a method for detecting changepoints in data. Binary Segmentation is a method which identifies the location of a potential changepoint on an interval and continues recursively on the intervals to the left and right of this location. This was first introduced in Vostrikova (1981) with consistency results proved in Ventrakaman (1991). Recently improved consistency results have been proved see Fryzlewicz (2014) for instance. We note that there are a number of alternatives to Binary Segmentation see Fryzlewicz (2014) for a good exposition (and another changepoint procedure: Wild Binary Segmentation).

We then discuss the application of hypothesis testing in conjunction with Binary Segmentation. In order to consider the distribution of the statistics against which we base our hypothesis tests, we use resampling techniques. In particular we introduce the concept of the stationary bootstrap, see Politis and Romano (2008), which is a method for resampling dependent data. We show how to apply it to identify changepoints where the underlying data is dependent. This is particularly applicable to fMRI data as over time brain states are dependent. We then describe the NCPD algorithm in detail and propose a modification to it that we have termed bootstrapping everything in order to avoid misidentification of changepoints.

Section 3 discusses the hypothesis testing trees that arise from using hypothesis testing in conjunction with Binary Segmentation. These trees occur when we have to reject a given hypothesis before we can reject a certain subset of other hypotheses and are very applicable to Binary Segmentation. In order to analyse these trees of hypotheses we need to consider methods for dealing with multiple testing corrections. We describe concepts such as the *FWER* and *FDR* and look at a series of methods for dealing with multiple hypotheses.

We then discuss the problem of *Sequential Hypothesis Testing* where you are given a sequence of m null hypotheses and you have to reject the first $\hat{k}$ for some $\hat{k} \leq m$. *Sequential Hypothesis Testing* is of great interest in general because it can be applied to decide what variables to include in a statistical model. There have been a number of approaches taken to it. The most competitive of these is *Forward.Stop* developed by G'Sell et al (2015). We explain how this works and describe a technique from Davison (2015) that turns independent hypothesis trees into sequences of hypothesis tests. The trees arising from Binary Segmentation are dependent and require different methods. In order to analyse these cases we develop a method: *TreeShuffle* which applies multiple hypothesis testing to a tree of hypotheses and then rearranges the set of rejections in the tree in order to reduce the number of false rejections.

Section 4 discusses a new problem which (to our knowledge) has not been considered in detail before. We call this problem *False then True*. It is a subcase of the Sequential Hypothesis Testing problem and assumes additionally that the first k null hypotheses are false and that the last $m - k$ are true for some unknown k. (Note that this extra assumption need not be true in the Sequential Hypothesis Testing case.) This problem is interesting in its own right and we discuss how methods for solving it can be used in biostatistics.

In order to investigate *False then True* we first consider the more general problem of identifying changepoints in distribution. This problem has been considered a bit in the literature (for example see Ross and Adams (2012) and Kifer et al (2004)). We develop a new algorithm we call *KSS* for identifying distribution changes. We then describe how *KSS* can be applied to the *False then True* problem by considering changepoints in the p-values. We then develop other methods for approaching the *False then True* problem and show that they control the *FDR*. We compare the performance of these methods and show that they are able to take advantage of the structure of the problem to be more powerful than *Forward.Stop* in the *False then True* context while still controlling the *FDR* to a given level.

Statistics has a large number of applications in neuroscience. In this essay we look at some of the theory behind these applications. We hope to have added a little to this already very diverse research topic. We very much enjoyed doing research in this area and hope you have just as much fun reading this essay as we have had writing it.

2 Changepoint Analysis

Changepoint analysis refers to the study of the existence and detection of changes in a given process. In this Section we will look at methods to identify both their number and location.

2.1 Changepoints

Changepoints or Structural Breaks are the points at which a given process changes in distribution. As an example, suppose we have the following deterministic process for $t \in \{1, \dots, T\}$, some $T \in \mathbb{N}$:

$$X_t = \begin{cases} \mu_1 & t \leq k \\ \mu_2 & t > k \end{cases}$$

for some $k \in \{1, ..., T\}$ and where $\mu_1 \neq \mu_2$. The above is a simple case of a step-change at $t = k$ (where k is unknown) from μ_1 to μ_2. Our task is to identify the location of the point k. In this situation it is easy to do, as we can just pick the point where the value changes. Everything becomes much more interesting if we add noise. More generally we can consider the model (as described in Fryzlewicz (2014)) for $t = 1, \dots, T$:

Model 1. *$X_t = f_t + \epsilon_t$ where f_t is some general piecewise constant function and ϵ_t is some noise term.*

Assumptions on f and ϵ will be specific to each situation. This process gives rise to the following natural definition:

Definition 1. Given the process X_t defined in Model 1, the **changepoints** of X_t are the points: $t = \eta_1, ..., \eta_N$, (some $N \in \mathbb{N}$) at which f_t changes value.

See Figure 1 below for examples of data which follow this model. In these examples all of the errors are normal and independent. (Note that in the top right panel of Figure 1 there is a changepoint, but it's not as noticeable as the others. Such changepoints are more difficult to identify.)

More generally, **changepoints** can refer to the locations of any arbitrary change in the data, which we then try to detect. Moreover there is no reason to restrict ourselves to 1-dimensional processes. Detection of changepoints in multivariate time series is an active research area, as is the investigation of change points in more complex data structures such as networks (as we shall see later in Section 2.2.6) or covariance matrices.

There are many different changepoint problems (we shall see quite a few examples in this essay). For now we shall restrict ourselves to Model 1 and examine some of its properties. This model is itself very rich. Indeed there have been many of papers written specific on it. For instance see: Yao and Au (1989), Lavielle and Moulines (2000). Fryzlewicz (2014) gives a comprehensive discussion of this. Many of the techniques which apply to this model can be extended to analyse different types of changepoint scenarios.

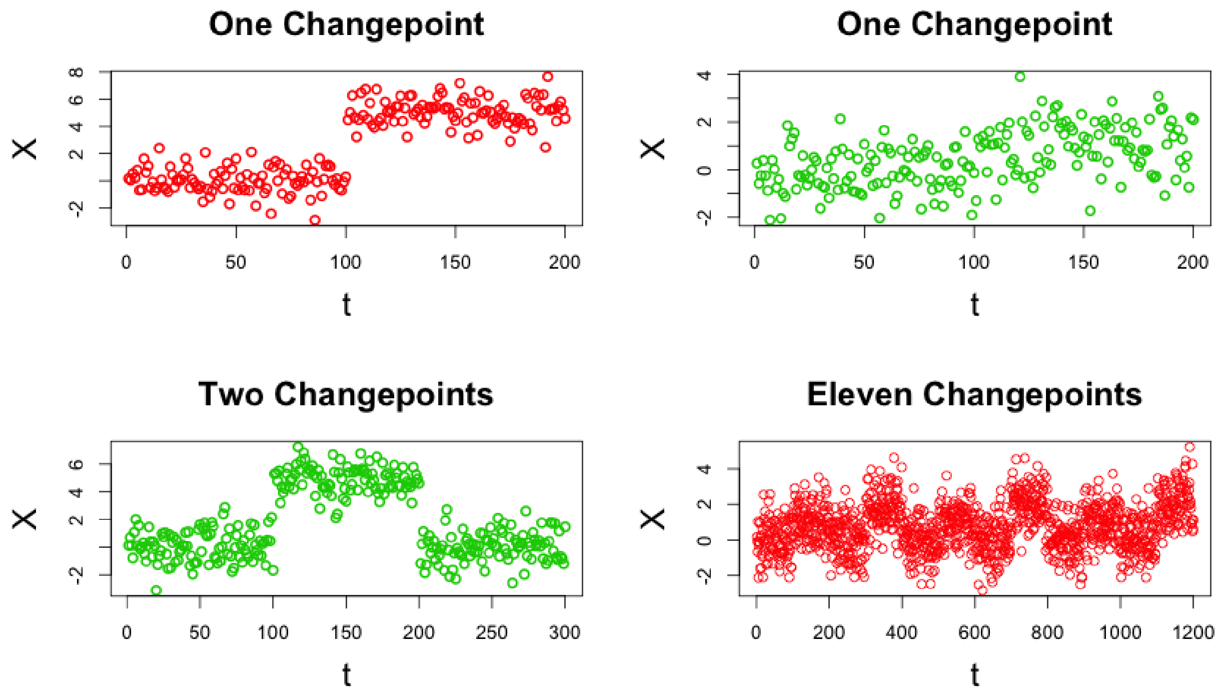


Figure 1: Examples of data following Model 1.

2.1.1 Binary Segmentation

In order to analyse the changepoints and their locations, we shall first look at the Binary Segmentation Algorithm. This algorithm identifies the most likely location of a changepoint in X on $\{1,\ldots,T\}$ and then repeats this on the subintervals to the left and right of the estimated changepoint. It keeps repeating on successive subintervals until some stopping criterion is satisfied. In order to apply this to Model 1 we require a means of identifying the difference between the points to a left and to the right of a potential changepoint.

Definition 2. In the case of Model 1, a commonly used measure of the difference at a point k is the CUSUM statistic (standing for Cumulative Sum), where for $e > s$, $n = e - s + 1$ and $k \in \{s, \ldots e-1\}$, this is defined as:

$$\tilde{X}^k_{s,e} = \sqrt{\frac{e-k}{n(k-s+1)}}\sum_{t=s}^{k} X_t - \sqrt{\frac{k-s+1}{n(e-k)}}\sum_{t=k+1}^{e} X_t$$

(Note the weights on the sums above are for technical reasons in the proof of consistency; see Lemma A.3 in Fryzlewicz (2014)). Intuitively we can consider this to be an adjusted difference in the means of the points to the right and to the left of the changepoint, which makes sense given the nature of our model. Then the method itself is quite simple, it finds the point at which the absolute value of the CUSUM statistic is maximal and then divides the data at that point. It then repeats the process again separately on the points to the left and to the right of this maximum. This algorithm is described formally below in Algorithm 1. Here BS$(\xi, 1, T)$ is what one should implement in order to the start the Algorithm.

See Figure 2 for an example of implementing Binary Segmentation with the CUSUM statistic. In this Figure, we have generated $X_t \sim N(0,1)$ for $1 \leq t \leq 100$ and $X_t \sim N(5,1)$ for $101 \leq t \leq 200$, with the changepoint occurring at 100 and have shown this in the top left panel. In the top right we see the CUSUM statistic evaluated on the whole interval. The panels at the bottom are the CUSUM statistics calculated to the left and the right of the changepoint. Note that in these bottom graphs there is no clear point at which there is a change. This is because there is only one changepoint in this example. Note also that even in these graphs the CUSUM statistic is maximized somewhere by random chance. This happens even though there is no changepoint to the left and right; hence the need for a hypothesis test.

Algorithm 1 Binary Segmentation applied to data $X = (X_1, \ldots, X_T)$

```
1: procedure BS($\xi, s, e$)
2:     Changepoints ← $\emptyset$
3:     if $e - s < 1$ then
4:         return Changepoints
5:     end if
6:     $C \leftarrow \{s, \ldots, e-1\}$
7:     $k \leftarrow \text{argmax}_{b \in C}\{|\tilde{X}^b_{s,e}|\}$
8:     if $|\tilde{X}^k_{s,e}| > \xi$ then
9:         Changepoints ← BS($\xi, s, k$) $\cup$ $\{k\}$ $\cup$ BS($\xi, k+1, e$)
10:    end if
11:    return Changepoints
12: end procedure
```

2.1.2 Binary Segmentation with Hypothesis Testing

In Algorithm 1, instead of choosing to add k to the set of changepoints when $|\tilde{X}^k_{s,e}| > \xi$, we can instead use a Binary Segmentation hypothesis testing procedure, where for each candidate changepoint k we consider the hypothesis testing pair:

$$H_0 : \text{There is no changepoint at } k \quad \text{vs} \quad H_1 : \text{There is a changepoint at } k$$

Here we add k to the set of changepoints if and only if the null hypothesis: H_0 is rejected. (See Algorithm 2: BShyp below.) For this BShyp$(1, T)$ is what one should implement in order to the start the Algorithm. In order to discuss this further, we need a means of testing this hypothesis and we shall use bootstrap resampling to do so; see Section 2.2.

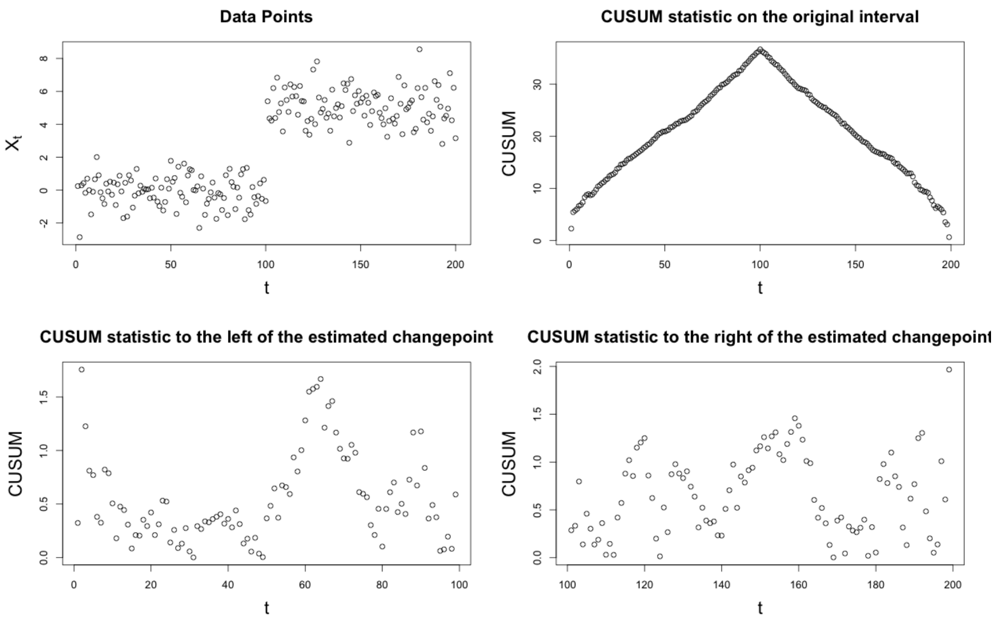


Figure 2: CUSUM statistic applied to a changepoint.

In BShyp we have evaluated the null hypotheses as we go along, another option is to save the hypothesis testing until the end. In other words to calculate a set of estimated changepoints using some Binary Segmentation procedure, and apply hypothesis testing to them only once they have all been calculated. We discuss this approach in Section 3.4. Regardless of which approach is taken, our procedures will require multiple hypothesis testing corrections to control the error rate (See Section 3 for details).

2.2 The Bootstrap and application to Changepoints

In the Binary Segmentation algorithm we would like to have a means of evaluating the null hypotheses at each stage. Ideally we could compare the observed value of the CUSUM statistic to its distribution. However this distribution is unknown and so we need to consider methods to estimate it. In this section we will consider the Bootstrap, a resampling method first developed by Efron (1979) designed to give a good estimate of the distribution of a function of the data. We will then consider the stationary bootstrap developed in Politis and Romano (1994) which enables us to resample dependent data. Having done this we will then look at how it can be applied in the context of Binary Segmentation.

As an example of how these procedures can be applied beyond the context of Model 1 we will then look at the NCPD Algorithm which identifies changes in resting brain states using fMRI data. In the context of fMRI data the stationary bootstrap will be very useful because the data we will be dealing with is certainly dependent. Here we consider the stationary bootstrap, however there are other methods for resampling dependent data such

Algorithm 2 Binary Segmentation applied to data $X = (X_1, \ldots, X_T)$

```
1:  procedure BSHYP($s, e$)
2:      if $e - s < 1$ then
3:          return
4:      end if
5:      $C \leftarrow \{s, \ldots, e-1\}$
6:      $k \leftarrow \text{argmax}_{b \in C}\{|\tilde{X}^b_{s,e}|\}$
7:      Consider the null hypothesis: $H_0$ that there is no changepoint at $k$.
8:      if $H_0$ is rejected then
9:          Changepoints $\leftarrow$ BS($s, k$) $\cup$ $\{k\}$ $\cup$ BS($k+1, e$)
10:     else
11:         Changepoints $\leftarrow \emptyset$
12:     end if
13:     return Changepoints
14: end procedure
```

as the moving-blocks bootstrap; see Kunsch (1989).

2.2.1 IID Bootstrap

The way that the bootstrap works is given observed values: $X_1, \ldots, X_T \overset{iid}{\sim} F$, we generate a pseudo-sample: $X^*_1, \ldots, X^*_T$, where (conditional on the observed values) each X^*_i is chosen uniformly from the set: $\{X_1, \ldots, X_T\}$, independently of the other $X^*_j, j \neq i$. In other words, we draw T i.i.d copies with replacement from: $\{X_1, \ldots, X_T\}$.

In practise we actually generate a number of pseudo-samples: B (all independent of each other, conditional on the observed values) using the above procedure. We will label the ith such pseudo-sample: $X^*_{i,1}, \ldots, X^*_{i,T}$. The idea of the bootstrap is that under the assumption that the data are i.i.d, we can treat the pseudo-samples as being samples from the original distribution. Thus given some function: $R = R(X_1, \ldots, X_T)$ we can perform inference on the distribution R using the bootstrapped samples. For $i = 1, \ldots, B$, let $R^*_i = R(X^*_{i,1}, \ldots, X^*_{i,T})$ then we can create an estimated $100(1-\alpha)\%$ confidence interval for $R = R(X_1, \ldots, X_T)$ using the empirical quantiles of $R^*_1, \ldots, R^*_B$. This confidence interval tends to a true $100(1-\alpha)\%$ confidence interval of R as B and T tend to ∞ (see Efron (1979) for details of the consistency arguments). This is an intriguing result since given the original sample, at first glance it seems like we only know the value: $R(X_1, \ldots, X_T)$ and cannot estimate its distribution.

2.2.2 Stationary Bootstrap

Having described the bootstrap, we now want to consider a resampling method which doesn't require the data to be independent, namely the stationary bootstrap. This requires that the data $(X_1, \dots, X_T)$ is strictly stationary and weakly dependent. The method samples blocks of the data where the lengths of the blocks are chosen according to the geometric distribution with parameter p and their starting points are chosen uniformly. The data is wrapped around to allow for starting points towards the end of the data-set, or blocks which are long enough to require this.

In order to describe it mathematically, let us first define: $D_{i,b} = \{X_i, X_{i+1}, \dots, X_{i+b-1}\}$, where given $j > N$, we define $X_j = X_i$ where $j \equiv i \pmod N$ and $1 \leq i \leq N$ and we take $X_0 := X_T$. Now (independently of the data) we generate $I_1, I_2, \dots,$ i.i.d chosen uniformly from $\{1, \dots, T\}$. Independently of the data and of the I_j, generate $L_1, L_2, \dots,$ i.i.d from the Geom(p) distribution. Then our generated pseudo-sample: $\{X_1^*, \dots, X_T^*\}$ is the first T observations of:

$$\{D_{I_1,L_1}, D_{I_2,L_2}, \dots, \}$$

As with the bootstrap we actually generate a number of pseudo-samples: B (all independent of each other, conditional on the observed values) using the above procedure. We will label the ith such pseudo-sample: $X_{i,1}^*, \dots, X_{i,T}^*$. As with the bootstrap, given some function: $R = R(X_1, \dots, X_T)$ we can perform inference on R using the pseudo-samples. For $i = 1, \dots, B$, let $R_i^* = R(X_{i,1}^*, \dots, X_{i,T}^*)$ then we can create an estimated $100(1-\alpha)\%$ confidence interval for $R = R(X_1, \dots, X_T)$ using the empirical quantiles of $R_1^*, \dots, R_B^*$. We require $B, T \to \infty$ for convergence, see Politis and Romano (1994) for details of the consistency arguments. They prove consistency results when taking R to be the mean and for smooth functions of the mean.

Note that the choice of p is related to the choice of the size of the block in the moving blocks bootstrap. For stationary bootstrap, the average blocksize is $1/p$ because the length of each block follows a geometric distribution.

2.2.3 Applying the Stationary Bootstrap to Changepoints

The stationary bootstrap can be used to calculate confidence intervals for a given statistic of the data. In section 2.1.2 we discuss the use of a Binary Segmentation hypothesis testing procedure. In this case for each given interval, we want to test whether there is a changepoint and to do this we use the CUSUM statistic and use the stationary bootstrap procedure to determine whether its value is significant.

On each interval: $[s, e]$, we take $k \leftarrow \text{argmax}_{b \in \{s,\dots,e\}}\{|\tilde{X}_{s,e}^b|\}$ and we define $\tilde{X}^{max} = |\tilde{X}_{s,e}^k|$. The stationary bootstrap can be used as a means of calculating a confidence interval for the value of $\tilde{X}^{max}$ and determining whether or not the observed value is what we would expect under the null hypothesis. What the stationary bootstrap effectively does is to reorder the data in such as manner that the dependence is not lost. Under the null hypothesis that there is no changepoint, shuffling the data around should not affect the distribution of $\tilde{X}^{max}$, so the observed value should lie in a $100(1-\alpha)\%$ confidence interval $100(1-\alpha)\%$ of the time. However if there was a changepoint in the data, then (assuming that the changepoint were identified correctly) the distribution of $\tilde{X}^{max}$ will change and we would expect the observed value to be significant.

See Figure 3 below for an example of applying the stationary bootstrap to data with and without a changepoint. In the top left panel we have taken $X_t \overset{i.i.d}{\sim} N(0,1)$ for $1 \leq t \leq 200$ (meaning that there is no changepoint) and we have applied the stationary bootstrap to this. As we expect the result (top right panel) is very similar to that of the original data. In the lower panels we do the same thing for data: Y however we have $Y_t \overset{i.i.d}{\sim} N(0,1)$ for $1 \leq t \leq 100$ and $Y_t \overset{i.i.d}{\sim} N(5,1)$ for $1 \leq t \leq 200$. Applying the stationary bootstrap to Y we can clearly see the effects of the random variables: I and L. As we can see the result is clearly different than the observed data. Note that we have taken $p = 200^{-1/3}$ and $B = 1000$ in our application of the stationary bootstrap.

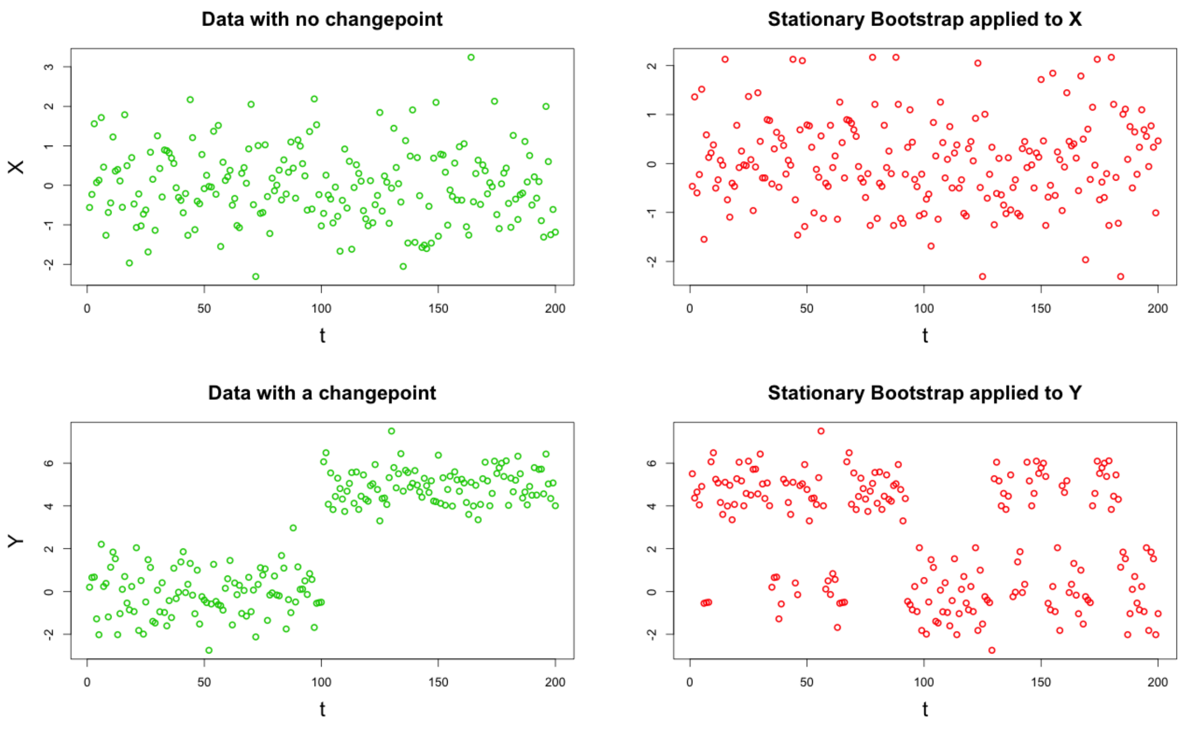


Figure 3: Applying the Stationary Bootstrap

To apply this to Binary Segmentation we find the location where the CUSUM is maximized. We then calculate a number of stationary bootstrap samples and for each of them recompute the test-statistic at the location of the original maximum. Then we calculate the

$100(1-\alpha)\%$ quantile of the set of samples (for some choice of α) and, if our observed: $\tilde{X}^{max}$ lies outside this interval, then we reject the null. If we reject the null, then we repeat the algorithm on the subintervals to the left and right of the estimated changepoint. We describe this procedure in detail in Algorithm 4 below. We have separately described the algorithm for computing the confidence interval using the stationary bootstrap in Algorithm 3 below.

Algorithm 3 Computing confidence levels for CUSUM with the stationary bootstrap

```
 1: procedure CUSUMboot($s, e, k, B$)
 2:     $n \leftarrow 1$
 3:     Use the stationary bootstrap to compute a new pseudo-sample from: $X_s, \ldots, X_e$
        (independent of all previous pseudo-samples) and call this: $X_s^*, \ldots, X_e^*$
 4:     Compute: $\tilde{X}_n := \left|\tilde{X}_{s,e}^k\left(X_s^*, \ldots, X_e^*\right)\right| = \left|\sqrt{\frac{e-k}{n(k-s+1)}}\sum_{t=s}^k X_t - \sqrt{\frac{k-s+1}{n(e-k)}}\sum_{t=s}^e X_t\right|$
 5:     if $n = B$ then return the $100(1-\alpha)\%$ quantile of $\tilde{X}_1, \ldots, \tilde{X}_n$
 6:     else
 7:         $n \leftarrow n + 1$
 8:         goto step 3
 9:     end if
10: end procedure
```

Algorithm 4 Binary Segmentation with the stationary bootstrap

```
 1: procedure BSboot($s, e, B$)
 2:     Changepoints $\leftarrow \emptyset$
 3:     if $e - s < 1$ then return Changepoints
 4:     end if
 5:     $B \leftarrow \{s, \ldots, e-1\}$
 6:     $k \leftarrow \text{argmax}_{b \in B}\{|\tilde{X}_{s,e}^b|\}$
 7:     if $|\tilde{X}_{s,e}^k| >$ CUSUMboot($s, e, k, B$) then
 8:         Changepoints $\leftarrow$ BShyp($s, k$) $\cup$ $\{k\}$ $\cup$ BShyp($k+1, e$)
 9:     end if
10:     return Changepoints
11: end procedure
```

2.2.4 Bootstrapping Everything

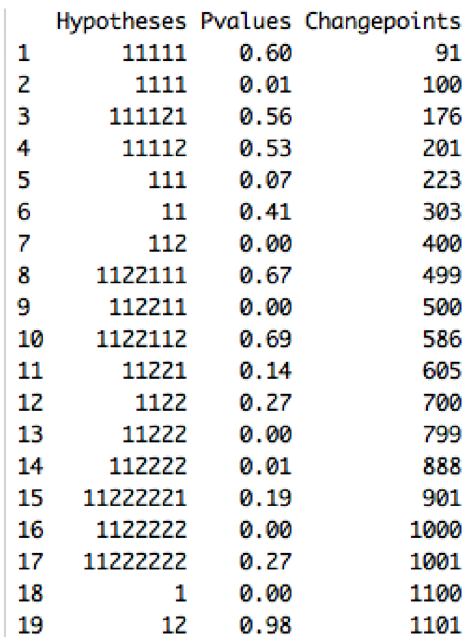
Now in the method above we have compared the observed value of $\tilde{X}^{max}$ to the value of the empirical distribution of the stationary bootstrapped $|\tilde{X}^k|$ (where $k \leftarrow \text{argmax}_{b \in \{s,\ldots,e\}}\{\tilde{X}_{s,e}^b\}$). In fact there is a potential problem with this. The issue is that even under the null hypothesis, there will be some value of j at which the CUSUM statistic is maximized, just by random chance. In computing the confidence interval, we have not taken into account the fact that we have chosen k because it was the the location of the maximum. Under the null, the chance of seeing significant values somewhere may be large but the change of seeing them at a specific place (ie k) is less likely. This means that we will have an oversignificance problem as the CUSUMboot Algorithm will return significant p-values even under the null hypothesis. So using it will lead us to identify changepoints when they do not exist. In order to get around this problem we should include the selection of k in our bootstrap procedure. In other words for each bootstrap sample, recompute the CUSUM statistic at each point and take the maximum value to be the bootstrapped value. This is formally described below in Algorithm 5: CUSUMbootall.

We can incorporate CUSUMbootall into Algorithm 4 by replacing CUSUMboot used in line 6 with CUSUMbootall (see below) (ie replace: $\tilde{X}_{s,e}^k >$ CUSUMboot(s, e, k, B) with $\tilde{X}_{s,e}^k >$ CUSUMbootall(s, e, B) in line 6). This will give us a new Binary Segmentation algorithm which eliminates the problem.

See Figure 4 for an example of the Binary Segmentation procedure in action under the use of the CUSUMbootall algorithm. The underlying data is the eleven changepoint example shown in Figure 1. Here the p-values are calculated as you go down the tree. Note that the true changepoints are at multiples of 100. We note that the locations of the changepoints could be further improved by applying a midpoint correction, see Appendix 5.

Algorithm 5 Computing confidence levels for CUSUM with the stationary bootstrap

```
1: procedure CUSUMBOOTALL($s, e, B$)
2:     $n \leftarrow 1$
3:     Use the stationary bootstrap to compute a new pseudo-sample from: $X_s, \ldots, X_e$
       (independent of all previous pseudo-samples) and call this: $X_s^*, \ldots, X_e^*$
4:     for $j = 1, \ldots, T$
5:         Compute: $\tilde{X}_n^j := \left|\tilde{X}_{s,e}^j\left(X_s^*, \ldots, X_e^*\right)\right| = \left|\sqrt{\frac{e-j}{n(j-s+1)}} \sum_{t=s}^{j} X_t^* - \sqrt{\frac{j-s+1}{n(e-j)}} \sum_{t=j+1}^{e} X_t^*\right|$
6:     end for
7:     $K \leftarrow argmax_{j=1,\ldots,T}\{\tilde{X}_n^j\}$; $\tilde{X}_n \leftarrow \tilde{X}_n^K$
8:     if $n = B$ then return the $100(1-\alpha)\%$ quantile of $\tilde{X}_1, \ldots, \tilde{X}_n$
9:     else
10:        $n \leftarrow n + 1$
11:        goto step 3
12:    end if
13: end procedure
```



```
   Hypotheses Pvalues Changepoints
1       11111    0.60           91
2        1111    0.01          100
3      111121    0.56          176
4       11112    0.53          201
5         111    0.07          223
6          11    0.41          303
7         112    0.00          400
8     1122111    0.67          499
9      112211    0.00          500
10    1122112    0.69          586
11      11221    0.14          605
12       1122    0.27          700
13      11222    0.00          799
14     112222    0.01          888
15   11222221    0.19          901
16    1122222    0.00         1000
17   11222222    0.27         1001
18          1    0.00         1100
19         12    0.98         1101
```

Figure 4: Applying the CUSUMbootall Algorithm

2.2.5 Comparing CUSUMbootall and CUSUMboot

To show that there is a difference between CUSUMbootall and CUSUMboot we can compare their performance under the null hypothesis that there is no changepoint. In order to do so we have independently generated 500 sequences of length 100 where each sequence has the same distribution as $X = (X_1, \ldots, X_{100})$ where $X_i \overset{iid}{\sim} N(0,1)$ for $1 \leq i \leq 100$. (So that there is no changepoint.) Then we have calculated the p-values obtained by implementing both the CUSUMboot and the CUSUMbootall methods on each of these sequences. The following definition will be helpful:

Definition 3. For $X_1, \ldots, X_n \overset{iid}{\sim} F$, we define the **estimated CDF** (ECDF) to be: $\hat{F}(x) = \frac{1}{n}\sum_{i=1}^{n} 1\left[X_i \leq x\right]$.

Thus implementing each Algorithm on the 500 sequences, independently, allows us to create an estimate of the distribution of their outputs by plotting the resulting ECDFs. We have done this for CUSUMboot in Figure 5 and for CUSUMbootall in Figure 6. For each implementation we have taken $B = 10,000$ so that we are taking $10,000$ stationary bootstrap iterations. Note that in this example the data is independent, of course the stationary bootstrap will certainly work in this situation.

From these figures we can see that when using the CUSUMboot algorithm, we have a definite over-significance problem, with the p-values tending to take significant values even though there is no changepoint in the data. Under the CUSUMbootall algorithm there is no over-significant bulge in the ECDF. In fact we have the Theorem 1, see below.

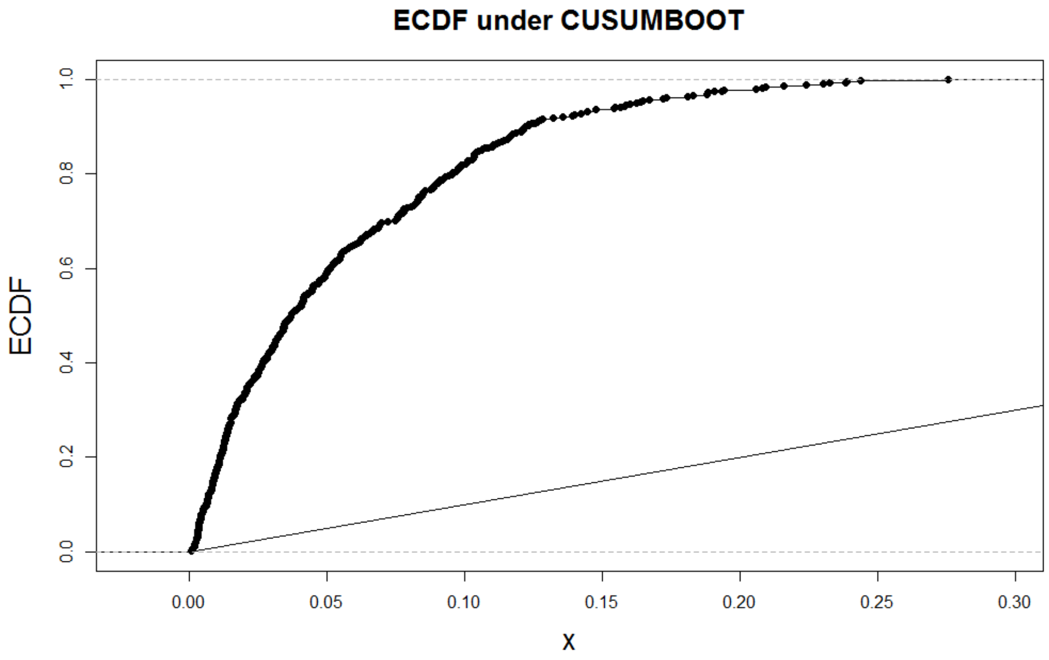


Figure 5: The ECDF of the p-values generated by CUSUMboot when there is no changepoint.

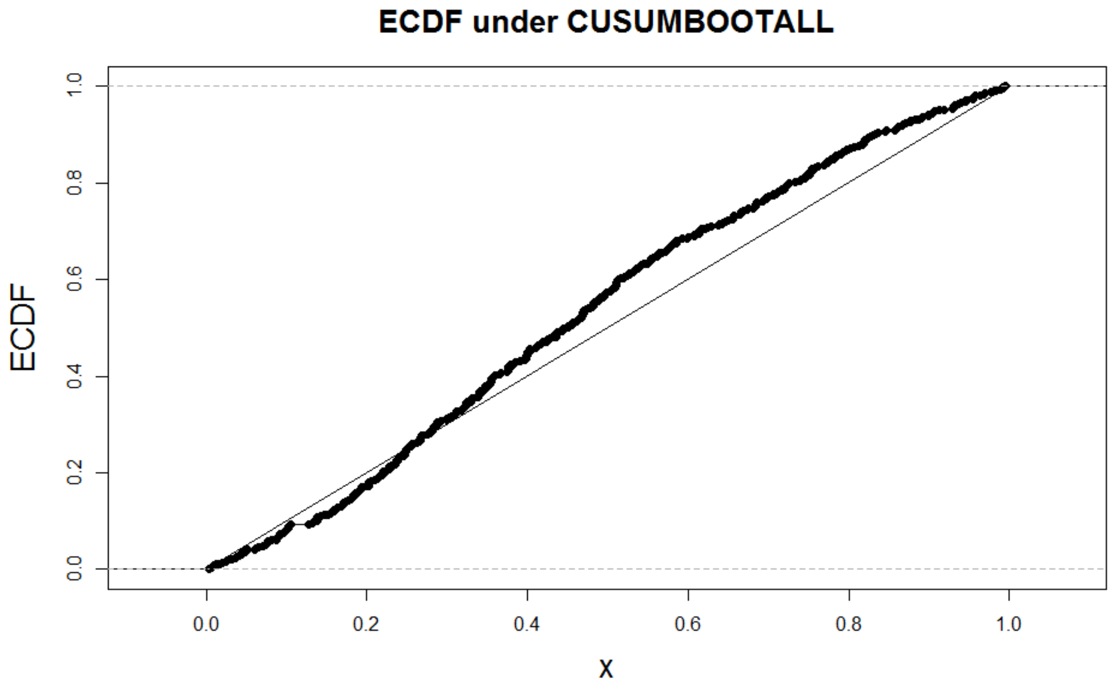


Figure 6: The ECDF of the p-values generated by CUSUMbootall when there is no changepoint.

Theorem 1. *Assume the X_i are i.i.d and have continuous CDF, then the CDF of $max_{k\in\{s,\dots,e\}}\{|\tilde{X}_{s,e}^k|\}$ is continuous. (Recall $\tilde{X}_{s,e}^k = \sqrt{\frac{e-k}{n(k-s+1)}}\sum_{t=s}^k X_t - \sqrt{\frac{k-s+1}{n(e-k)}}\sum_{t=s}^e X_t$.)*

Proof: See Appendix A for Lemmas $4, 5, 6, 7$. Given these lemmas the proof following quite easily. Multiple applications of Lemma 4 imply that the CDF of $\tilde{X}_{s,e}^k$ is continuous and then applying Lemma 6 and Lemma 7 yields the result. □

The data in our example is normal so the CDF of X_i is continuous for all i. Thus applying Theorem 1 combined with Theorem 2 from Section 3.2 we would expect the p-values to be uniform under the null hypothesis, assuming we could evaluate their true distribution. We see that this seems reasonable from the CUSUMbootall graph. Of course we would only expect the ECDF to converge to the uniform distribution as B and T tend to infinity; as convergence for the stationary bootstrap occurs in the limit.

To further illustrate the problem of over-significance, we can consider a simple changepoint problem where $X_t \sim N(0,1)$ i.i.d for $1 \leq i \leq 100$ and $X_i \sim N(0,5)$ i.i.d for $1 \leq i \leq 100$. Applying Binary Segmentation with the CUSUMboot algorithm tends to find around 3 changepoints where in fact there is only one. Applying the CUSUMbootall algorithm works much better and in simulations correctly finds that there is only one changepoint as well as giving a good estimate for its location. For this example we have not taken into account the need for multiple testing corrections, as we have just considered rejecting each test at the 0.05 level. See Section 3 for a discussion of multiple testing applied to Binary Segmentation. Nevertheless, this provides an illustration of why the CUSUMbootall algorithm is to be preferred.

2.2.6 Network Changepoint Detection

Now we will look to apply some of the techniques we have been considering above in the CUSUM case to a more complicated example. The algorithm that we will consider here is Network Changepoint Detection (NCPD), (developed recently in Cribben and Yu (2016)) which is an algorithm for detecting changepoints in the community structure of a multivariate time series network. Here we'll provide an overview so that we can apply the changepoint techniques we have developed to the NCPD algorithm, but we refer the interested reader to Cribben and Yu (2016) for more details.

In the change in mean changepoint scenario we have seen above we used the CUSUM statistic to measure the difference between the data to the left of a potential changepoint and the data to the right of it. For Network Changepoint Detection, we have to use a different statistic which measures the difference in the network structure instead the difference in mean. To do this, Cribben and Yu (2016) use a technique called Spectral Clustering to estimate the community structure of the network to the left and right of a potential changepoint.[1] This generates N by K matrices $U_{L,k}$ and $U_{R,k}$. Here N is the number of nodes in the network and the ith row of $U_{L,k}$ is the centroid of the community that the ith node belongs to on the left hand side of the potential changepoint: k (estimated using the data to the left of k). $U_{R,k}$ is similarly defined for the right hand side. See Von Luxburg (2007) for more details about Spectral Clustering and Newman (2004) for more details about Community Detection in networks.

Having defined $U_{L,k}$ and $U_{R,k}$, Cribben and Yu use this to create a statistic: γ_k at each point k which identifies the differences between the data to the left and to the right of k. This is defined as:

Definition 4. $\gamma_k = \sum_{i=1}^{K} \sigma_i$ where the σ_i are the singular values of $U_{L,k}^T U_{R,k}$

We can then combine this with the stationary bootstrap to develop the NCPD algorithm, see Algorithm 6 below. Note the minimization in the calculation of k in line 8, this is because the lower the value of γ the greater the difference, unlike with the CUSUM statistic. Note

Algorithm 6 Network Changepoint Detection

```
1: procedure NCPD($Y, B$)
2:     Changepoints ← $\emptyset$
3:     $T \leftarrow length(Y)$
4:     if $T < 2n_{min}$ then
5:         return Changepoints
6:     end if
7:     $C \leftarrow \{n_{min}, \dots, T - n_{min}\}$
8:     At each point $t \in C$, set $\gamma_t$ to be the value of $\gamma$ at point $t$.
9:     $k \leftarrow \text{argmin}_{t \in C}\{\gamma_t\}$
10:    Apply the stationary bootstrap to resample the data: $Y$ (independently) $B$ times.
11:    Calculate a value for $\gamma_k$ for each resampled dataset.
12:    if the observed value of $\gamma_k$ is significant then
13:        Changepoints ← NCPD($Y[1,k], B$) $\cup$ $\{k\}$ $\cup$ NCPD($Y[k+1,T], B$)
14:    end if
15:    return Changepoints
16: end procedure
```

that calculating an estimate of a network requires a certain number of data points. In our presentation of the algorithm we have denoted the minimum amount required as n_{min}.

[1]Assuming that there are K communities for some K. Optimal choice of K is an open question see Chen and Lei (2014) and Franco Saldaña et al (2014). The choice of K is discussed in Cribben and Yu (2016) where they note that their algorithm is robust under simulation to the choice of K so long as it is over-estimated rather than under-estimated.

This procedure will result in a hypothesis testing tree, analogous to the one obtained by applying CUSUM methods to the mean change problem. See Section 3.3 and 3.4 for details of dealing with trees of hypotheses.

We note that the NCPD algorithm also specifies a means for dealing with outliers in the γ values but we have not gone into this here (see Cribben and Yu (2015) for details). Also note that above we use an analogue to the CUSUMboot algorithm, in that for each bootstrap sample we calculate the γ value at the potential changepoint. In light of our discussions above, we would recommend modifying the NCPD algorithm to incorporate the concept of bootstrapping everything. This would involve introducing an analogue of CUSUMbootall to determine whether an observed value is significant. This modification will eliminate the problem of over-significance and avoid any misidentification of changepoints.

3 Hypothesis Testing

As discussed in the previous chapter, Binary Segmentation often results in a number of null hypotheses which we would like to analyse. In order to do so we have to apply some multiple hypothesis testing correction. The null hypotheses resulting from Binary Segmentation are nested and we can change our multiple testing procedures in order to take advantage of this.

3.1 Hypothesis Tests and p-values

First we define what we usually mean by a hypothesis testing problem. Let $\mathbf{Y} = (Y_1, \dots, Y_n)$ be a vector of i.i.d real random variables each with cumulative density function: $F(x) = \mathbb{P}(Y_1 \geq x)$ which may be unknown. Then in hypothesis testing, given observed data: $\mathbf{y} = (y_1, \dots, y_n)$, we want to decide between two hypotheses about F, which we call H_0 (the **null hypothesis**) and H_1 (the **alternative hypothesis**). We will refer to the pair: H_0, H_1 as a **hypothesis pair**. We can then define a **hypothesis test** to be a method that partitions $\mathbb{R}^n$ into two regions: $\mathcal{R}$ and $\mathcal{R}^C$ and rejects H_0 in favour of H_1 if $\mathbf{y} \in \mathcal{R}$ and does not reject H_0 if $\mathbf{y} \in \mathcal{R}^C$. We call $\mathcal{R}$ the **rejection region**. When we are referring to hypothesis tests it is also helpful to define the two types of error that can occur:

Definition 5. For a given hypothesis test, we say that a **Type 1 error** occurs if we reject H_0 when in fact H_0 is true. A **Type 2 error** occurs if we do not reject H_0 when in fact H_0 is false. Correspondingly, we define: $\mathbb{P}(\text{ Type 1 error }) = \mathbb{P}(\mathbf{Y} \in \mathcal{R} \mid H_0 \text{ is true})$ to be the **size** of the test and the **power** of the test to be $1 - \beta$ where $\beta = \mathbb{P}(\text{ Type 2 error }) = \mathbb{P}(\mathbf{Y} \notin \mathcal{R} \mid H_0 \text{ is false})$.

Typically in hypothesis testing we seek tests which have low probabilities of errors; we aim to control the size and subject to this control we look to maximize the power. To analyse this, it's often helpful to define a test statistic which is some real function of the data: $K = K(\mathbf{Y})$. This can be used for inference about the hypothesis test when we have knowledge about its distribution under the null hypothesis. Another concept that will be useful is that of the p-value:

Definition 6. Given a test statistic $K : \mathbb{R}^n \to \mathbb{R}$, suppose that K has (known) CDF: G under the null hypothesis.[2] Given an observed value of the test statistic: K^*, we define the **p-value** to be the probability under the null hypothesis of the test statistic taking the value K^* or a value more extreme than K^*. For a left tailed event this value is: $G(K)$ and for a right tailed event this value is $1 - G(K)$.

[2]Note when H_0 is a composite hypothesis it may not be the case that the CDF of T is a given function (as there may not be a unique CDF). In that case we in fact no longer have that the p-values are uniformly distributed, however they are still stochastically dominated by the uniform distribution. We will only consider cases where H_0 is a simple hypothesis so this is not an issue.

It is important to note that the p-value is a random variable itself and has a given distribution under each of the hypotheses. Indeed we can say something about its distribution under the null hypothesis. The following results will prove to be very useful (see Section 4.2).

Lemma 1. *Let G be a continuous CDF and let $X \sim G$. Then $G(X) \sim U[0,1]$.*

Proof: Let $\phi(x) = \mathbb{P}(G(X) \leq x)$. Given $x \in (0,1)$, choose $\epsilon > 0$ such that $x + \epsilon < 1$. G is continuous so for all $r \in [0,1]$ there exists $a_r \in \mathbb{R}$ such that $G(a_r) = r$ by the intermediate value theorem as $G(0) = 0$ and $G(1) = 1$. Taking $b_n = a_{x+\epsilon/n}$ and using the fact that G is increasing, we have that:

$$\{X \leq a_x\} \subseteq \{G(X) \leq x\} \subseteq \{X \leq b_n\}$$

$$\implies \mathbb{P}(X \leq a_x) \leq \mathbb{P}(G(X) \leq x) \leq \mathbb{P}(X \leq b_n) \text{ so } G(a_x) \leq \phi(x) \leq G(b_n)$$

The above holds for all $n \in \mathbb{N}$ and for all $x \in U[0,1]$. It follows that $\phi(x) = x$ for all $x \in [0,1]$ by continuity of G (as taking $n \to \infty$, $G(b_n) \to x$) and so $G(X) \sim U[0,1]$. □

Theorem 2. *Given a hypothesis test and an associated test-statistic K, with known continuous CDF G under the null hypothesis, the associated p-value p is uniformly distributed.*

Proof: We have $p = G(K)$ or $p = 1 - G(K)$ depending on the type of event and so $p \sim U[0,1]$ follows by the above lemma. □

Note that the conclusion of the above theorem is false if G is not continuous. However, we can still prove something in this case, see Theorem 3 below. This requires the following lemma:

Lemma 2. *Let G be a CDF and let $X \sim G$. Then for all $x \in \mathbb{R}$, $\mathbb{P}\left(G(X) \leq x\right) \leq x$.*

Proof: G is a CDF so it is right continuous, so part of the argument from 1 above still holds. We still have $\mathbb{P}(G(X) \leq x) \leq \mathbb{P}(X \leq b_n)$ (as G is increasing) and so taking $n \to \infty$ implies that $\mathbb{P}(G(X) \leq x) \leq x$ using right continuity. □

Theorem 3. *Given a hypothesis test and an associated test-statistic: K, with known CDF G under the null hypothesis, the associated p-value p has $\mathbb{P}(p \leq x) \leq x$.*

Proof: We have $p = G(K)$ or $p = 1 - G(K)$ depending on the type of event and so this follows by the above lemma. □

We can then choose to accept or reject the null hypothesis based on the value that test-statistic takes; if it takes an extreme value then we should reject. If we reject $\iff p \leq \alpha$ for some α, then we will have controlled the Type 1 error to a level α. This holds as: $\mathbb{P}(\text{ Reject } H_0 \mid H_0) = \mathbb{P}(\, p \leq \alpha \mid H_0 \,) \leq \alpha$ by Theorem 3.

3.2 Multiple Hypothesis Testing

When we consider a number of hypothesis tests (instead of just one) it is no longer sufficient to consider each test individually and we need to introduce some sort of multiple testing correction. The reason for this is that, given a set of hypothesis tests, if we controlled the size of each of them to a level α, we would expect that a number of Type 1 errors would occur, just by random chance. For instance, given 1000 hypothesis tests where for each test the null hypothesis was true, if we controlled each at a level of 5 percent, then the expected value of the number of false rejections is 50. This is despite the fact that all the null hypotheses are true. As such, it has become common to look at controlling other statistics such as the FWER and FDR. We define these below, as well as describing several well-known methods for controlling them.

To frame the set-up of the problem, suppose we have m null hypotheses: $H_1, \dots, H_m$, of which m_0 are true and $m - m_0$ are false (in no particular order). Let them have associated p-values: $p_1, \dots, p_m$, and let N be the number of true null hypotheses which are rejected and let $I_0 \subseteq \{1, \dots, m\}$ be the set of indices of the true null hypotheses.

Definition 7. **The Familywise Error Rate** or **FWER** $= \mathbb{P}(N \geq 1)$ is the probability of rejecting at least one true null hypothesis.

One simple method for controlling the FWER is the Bonferroni correction, see Bonferroni (1936), which rejects each hypothesis $H_i \iff p_i \leq \alpha/m$. We then have the following result:

Theorem 4. *Under the Bonferroni correction, we have FWER* $= \mathbb{P}(N \geq 1) \leq \alpha$

$$\textit{Proof: } N \in \mathbb{N} \cup 0, \text{so we have: } \mathbb{E}(N) = \sum_{n=0}^{\infty} \mathbb{P}(N \geq n) \geq \mathbb{P}(N \geq 1)$$

$$\text{So if follows that } \mathbb{P}(N \geq 1) \leq \mathbb{E}(N) = \mathbb{E}\left(\sum_{i \in I_0} 1\left[p_i \leq \alpha/m\right]\right)$$

$$= \sum_{i \in I_0} \mathbb{P}(p_i \leq \alpha/m) \leq \frac{m_0 \alpha}{m} \leq \alpha$$

We used Theorem 3 for the middle inequality. □

Note the Bonferroni correction does not require any independence assumption on the p-values.

There are a number of other procedures for controlling the *FWER*, including for instance Holm's procedure see Holm (1979). Most multiple testing procedures until the 1990s aimed to control the FWER. However, Benjamini and Hochberg (1995) proposed an alternative less conservative statistic know as the **False Discovery Rate** or **FDR** defined as:

$$FDR \ = \mathbb{E}\left(\frac{N}{\max(R,1)}\right) \text{ (Where R is the total number of rejections.)}$$

The *FDR* was introduced because it is less conservative than the *FWER*. We can show this with the following theorem:

Theorem 5. *$FDR \leq FWER$*

Proof: $R = 0$ implies that $FDR = 0$, so it follows that:

$$FDR = \mathbb{E}\left(\frac{N}{\max(R,1)}\right) \leq \mathbb{E}\left[\frac{N}{R}\middle| R \geq 1\right]$$

$$= \mathbb{E}\left[\frac{N}{R}\middle| N = 0, R \geq 1\right]\mathbb{P}(N=0) + \mathbb{E}\left[\frac{N}{R}\middle| N \geq 1, R \geq 1\right]\mathbb{P}(N \geq 1)$$

$$= \mathbb{E}\left[\frac{N}{R}\middle| N \geq 1, R \geq 1\right]\mathbb{P}(N \geq 1) \leq \mathbb{P}(N \geq 1) = FWER$$

Where we use the fact that $N \leq R$ which implies that $\mathbb{E}\left[\frac{N}{R}\middle| N \geq 1, R \geq 1\right] \leq 1$. $\square$

In their paper they introduced the **Benjamini-Hochberg procedure** (we will often refer to this as the **BH procedure**) which orders the p-values as $p_{(1)}, \ldots, p_{(m)}$ and takes $\hat{k} = \max\left(i : p_{(i)} \leq \frac{i\alpha}{m}\right)$. The procedure then rejects $H_{(1)}, \ldots, H_{(\hat{k})}$ and accepts $H_{(\hat{k}+1)}, \ldots, H_{(m)}$. They then prove the following theorem:

Theorem 6. *Suppose that the values: p_i, $i \in I_0$ are independent, and independent of $\{p_i : i \notin I_0\}$. Then the BH procedure controls the FDR to a level α, in fact $FDR \leq \frac{\alpha m_0}{m}$.*

In fact the FDR of the BH procedure is equal to $\alpha m_0/m$ under the additional assumption that the p-values are U[0,1] distributed under the null hypothesis.

However notice that above we require an independence assumption on the p-values. In fact we can replace this condition with that of **positive dependence**, only requiring that:

$$\mathbb{E}\{\phi(p_1, \ldots, p_n) \mid p_i = u\} \text{ is non-decreasing in } u \text{ for each } i \in I_0$$

for any (coordinate-wise) non-decreasing function ϕ. Then we still have that the BH-procedure controls the FDR see Benjamini and Yekutieli (2001), Sarkar (2002, 2008b). This result will be useful to us, see Section 3.4.

To deal with further dependence, Benjamini and Yekutieli (2001) proposed the alternative: Benjamini - Yekutieli procedure which instead takes:

$$\hat{k} = \max\left(i : p_{(i)} \leq i\alpha/\gamma_m m\right) \text{ where } \gamma_m = \sum_{j=1}^{m} 1/j$$

This procedure rejects $H_{(1)}, \ldots, H_{(\hat{k})}$ and accepts $H_{(\hat{k}+1)}, \ldots, H_{(m)}$. This allows arbitrary dependence between the p-values while still controlling the FDR to a level α, indeed still maintaining FDR $\leq \frac{\alpha m_0}{m}$. However in order to allow for this dependence, the procedure is very conservative; while the bound on the *FDR* is α in reality the *FDR* tends to be much lower.

3.3 Sequential Hypothesis Testing and Trees

We now consider how to deal with the multiple hypotheses that arise from Binary Segmentation. This situation is different to that of usual multiple testing because our hypotheses are nested and we would like to take advantage of this. This nesting occurs because if we decide that there is no changepoint on a given interval then we don't then need to further consider subintervals for changepoints. So if we were to accept the null on a given interval we should stop the algorithm and not consider further changepoints on that interval. In order to illustrate this we introduce the concept of trees. First we will describe an approach to this problem which requires discussing Sequential Hypothesis Testing.

3.3.1 Sequential Hypothesis Testing

Consider the following problem which is that of Sequential Hypothesis Testing:

Problem 1. *We have null hypotheses: $H_1, H_2 \ldots, H_m$ and we must reject $H_1, H_2 \ldots, H_{\hat{k}}$ and accept $H_{\hat{k}+1}, H_{\hat{k}+2} \ldots, H_m$ for some $\hat{k} \in \{0, \ldots, m\}$.*

For this problem we will assume that each null hypothesis H_i has an associated p-value: p_i and that we are testing each null hypothesis against some alternative hypothesis. We then must seek a procedure which rejects the first $\hat{k}$ hypotheses and accepts the last $m - \hat{k}$ and which controls the *FDR*. Methods for dealing with this problem can be applied to variable selection problems. This is because these problem often involve a sequence of null hypotheses that have to be considered where the first set of null hypotheses have to be rejected and the last set accepted.

This situation is considered in G'Sell et al (2015), who propose the two criterions below and show (using a martingale argument) that they both control the FDR at a level α under the condition that the p-values are independent. (Indeed they show that Strong.Stop(α) controls the FWER to a level α). The two procedures are the following:

Forward.Stop(α) which takes $\hat{k} = \max\left\{k \in \{1, \ldots, m\} : -\frac{1}{k}\sum_{i=1}^{k} \log(1 - p_i) \leq \alpha\right\}$

Strong.Stop(α) takes $\hat{k} = \max\left\{k \in \{1, \ldots, m\} : \exp\left(\sum_{j=k}^{m} \frac{\log(1-p_j)}{j}\right) \leq \frac{k\alpha}{m}\right\}$

Note above we assume that $\max(\emptyset) = 0$, so that $\hat{k} = 0$. This corresponds to the case where no rejections are to be made.

3.3.2 False then True

We have considered the general Sequential Hypothesis Testing problem above. We will now introduce a subcase of this general problem which we have termed *False then True* which is still very interesting and applicable. The problem to be considered is:

Problem 2. ***False then True:*** *We have null hypotheses: $H_1, H_2 \ldots, H_m$, where we know that the first k are false and the last $m-k$ are true for some unknown k and we must reject $H_1, H_2 \ldots, H_{\hat{k}}$ and accept $H_{\hat{k}+1}, H_{\hat{k}+2} \ldots, H_m$ for some $\hat{k} \in \{0, \ldots, m\}$.*

As above we assume that each null hypothesis H_i has an associated p-value: p_i and that we are testing each null hypothesis against some alternative hypothesis. The key difference between this and Problem 1 is that here we assume that the null hypotheses are false and then true. The setting of Problem 1 is more general than this and allows for the mixing of the true and false null hypotheses.

Even though it is a subcase of the more general problem it is still very applicable. Let us consider an example of a situation in which it applies. Suppose that an individual undergoes tests for some disease at time periods $1, \ldots, T$. Where for each time we want to test the null hypothesis that the individual has a given disease against the alternative hypothesis that they do not and where we expect the individual to develop the disease at a certain unknown time. In this case we would want to identify the time that the disease occurs so that we can identify when to begin treatment. This would result in a False then True type structure.

In this particular case, we would probably want to begin treatment as soon as possible after the disease was discovered. This would require detecting the disease in an **online** fashion (as we are going along) so that it could be identified as soon as possible after it occurred. This scenario is most applicable when our tests for the disease are accurate only a fraction of the time as then repeated application of the test is required. This is because when the tests are not always accurate we have to do multiple tests. This leads to a sequence of tests (as we do them over time) where the null hypotheses are false and then true.

An example of this hypothesis testing situation in practise, that we have seen in Biostatistics, is that of using Angiography to measure the presence of Stenosis. This is an imperfect measure (see Heiserman (2005)) and so methods for solving Problem 2 will apply to this situation. Of course there are many other such examples, as many tests of diseases tend to be imperfect. More generally if we wanted to identify the time of some binary change, and have a test to measure this at a number of time periods both before and after the change, then this would result in the *False then True* problem.

3.3.3 Trees of Hypotheses

In order to discuss the multiple hypotheses arising from Binary Segmentation, it will be helpful to introduce the concept of a hypothesis testing tree. This is when we have a number of hypotheses which are arranged in a tree and we have to reject/accept them in an order imposed by the tree. This arises naturally from the Binary Segmentation algorithm because initially we test the null hypothesis that there is no changepoint on the interval $[1, T]$. If we reject this (and estimate a changepoint location of r, say), we then consider the subsequent null hypotheses on the intervals $[1, r]$ and $[r + 1, T]$. Continuing in this fashion results in a tree of hypothesis tests. (For our purposes we will mainly be interested in the case of trees which divide by two at each stage, because of their applications to the Binary Segmentation procedure. However of course one can naturally imagine extensions to multiple and irregular divisions at each stage.)

More formally a **hypothesis tree** consists of a directed acyclic graph (where each node is a null hypothesis which is to be tested against some alternative hypothesis) starting at a root node H_{root}. See Figure 7 below for an example of this.

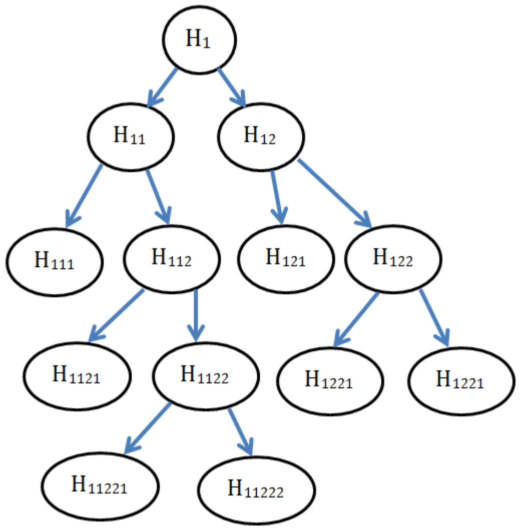


Figure 7: This is an example of a hypothesis tree.

Given a node H_A which has a directed edge to another node H_B we say that H_A **leads to** H_B and denote this by $H_A \to H_B$. Each node of the graph is either connected by directed edges to some other nodes or is a **leaf** in that it doesn't lead to other nodes. A **path** from H_A to H_B is a sequence of hypotheses: $H_1, \ldots, H_n$ such that $H_i \to H_{i+1}$ for $1 \leq i < n$ and where $H_1 = H_A$ and $H_n = H_B$. Given a path whose initial vertex is H_{root} (the root of the tree) and whose end vertex is a leaf, we refer to this path as a **branch**.

Given nodes H_A and H_B such that H_A leads to H_B by a path, we call H_B a **descendant** of H_A and we call H_A a **ancestor** of H_B. In the case where this path has length 1, we refer to H_B as a **child** of H_A and call H_A the **parent** of H_B. We refer to to the set of ancestors of a given hypothesis: H as $\mathcal{A}(H)$ and the set of descendants as $\mathcal{D}(H)$ and the parent of H (if it exists) as $\mathcal{P}(H)$.

The consideration of hypothesis testing as we go along requires us to use online methods. When implementing Binary Segmentation in Chapter 2 we applied the hypothesis testing at each stage; stopping the procedure on subintervals if we accepted a null hypothesis. However, we could also consider just maximizing the CUSUM statistic at each stage and only consider the hypothesis testing at the end (having reached some stopping criterion). This is the approach taken in the code of the alogorithm NCPD in Cribben and Yu in that they save the hypothesis testing until the end though they do not discuss a multiple testing correction. See Section 3.4 for methods for applying multiple correction methods to trees of hypotheses. We will be particularly interested in trees of hypothesis tests where each branch is in the form of Problem 2. Formally we can this define this problem via the following:

Problem 3. ***False then True Trees:*** *Given a hypothesis testing tree, we say the tree is* ***False then True*** *if for each branch: $\mathcal{B} = (H_1, \ldots, H_m)$, the first $k(\mathcal{B})$ null hypotheses are false and the last $m - k(\mathcal{B})$ are true for some unknown $k(\mathcal{B})$.*

As usual we want to identify which hypotheses are false and which are true. This particular problem is very applicable because any hypothesis tree arising from Binary Segmentation will be in this form.

3.3.4 Application of Sequential Hypothesis Testing to Trees

Given the False then True Tree problem, we want to reject the ones at the top of the tree which are false and accept those at the bottom which are true. Any method that we would consider should have the property that having accepted a given null hypothesis: H_j it also accepts all descendants of H_j. Given this situation, we could consider a mapping from the tree which would turn it into a Sequential Hypothesis Testing problem, so long as we ensure that all descendants of a given hypothesis appear after it in the mapping. Given a tree of null hypotheses, let us label them as: $H_1, \ldots, H_T$ (in no particular order) with associated p-values: $p_1, \ldots, p_T$.

Definition 8. For a given null hypothesis in the tree: H, define $p(H)$ to be its associated p-value.

Davison (2015) considers the following mapping from the tree of null hypotheses to a sequence: $H_1^s, \ldots, H_T^s$, where we:

1. Set H_1^s to be root of the hypothesis tree.

2. For $2 \leq r \leq T$, set $H^s_r = H_j$ where j is such that:
   $p_j \in \min\{p(H^s_i) : H^s_i$ is a child of one of $H^s_1, \ldots, H^s_{r-1}\}$
   If there is more than one j that satisfies this, one is selected at random.

Having performed this mapping to yield a sequence of hypotheses: $H^s_1, \ldots, H^s_T$ we seek to reject the first k for some k and accept the remainder. As such we can apply procedures for dealing with the Sequential Hypothesis Testing Problem (such as *Forward.Stop*) in order to select which hypotheses to accept and reject.

However, there is actually an issue: we cannot apply the Forward Stop or Strong Stop methods above and still be certain of controlling the FDR at a level α because these methods require that the p-values are independent. In the trees derived from Binary Segmentation scenario we do not have this independence. This is because some of the data we use to evaluate a given hypothesis will also be used in the evaluation of each of its descendant hypotheses. This overlap of data means that there will be some sort of dependence between the hypotheses. (Note this is true only for Binary Segmentation and not for False then True trees in general.)

3.4 Hypothesis Testing in Trees

Ideally we would like to develop new algorithms to deal with the Sequential Hypothesis Testing Problem under dependence. Having done this we could then control the *FDR* in the case of trees, after applying the above mapping from Section 3.3.4. Unfortunately this is still an open problem. We can take advantage of the False then True type structure that arises from the trees resulting from Binary Segmentation to develop algorithms that control the *FDR* in trees without using the sequential mapping above. We will show that these methods can be applied to both independent and dependent trees.

3.4.1 *TreeShuffle*

Suppose we have a multiple testing procedure: *Mhyp*, such that under some assumptions *Mhyp* controls either the *FDR* or the *FWER*. So long as the p-values in our tree obey the same assumptions as *Mhyp*, we can apply *Mhyp* to them. Doing this will control the *FDR* in the tree just as in the case where there are multiple hypothesis testing scenario. However, we can use the structure of the tree to improve on this by shifting the rejections to the top. More formally we describe this via the method *TreeShuffle* see Algorithm 7. To discuss this it will be helpful to define the following:

Definition 9. Suppose we have a tree $\mathcal{T}$ and each null hypothesis in this tree has either been accepted or rejected. Then we define the **Rejection Set:** $\mathcal{R}$ to be set of null hypotheses which have been rejected.

TreeShuffle is to be applied after we have used a multiple testing method to yield a rejection set: $\mathcal{R}$. Essentially what it does is to move the rejections, which will be scattered around the tree, to the top of the tree so that we reject is such a manner as to solve Problem 3. A nice analogy for it is to consider the rejections to be air bubbles and the others to be water drops. Then on applying our procedure, the water will flow to the bottom of the tree while the air goes to the top.

See Figure 8 for a tree after a multiple hypothesis testing procedure has been applied. Here we have $\mathcal{R} = \{H_1, H_{11}, H_{122}, H_{1121}, H_{1221}, H_{11221}\}$.

See Figure 9 for the result of applying *TreeShuffle* to this tree. Notice that in Figure 9 we have included two possible applications of *TreeShuffle*, based on different initial choice of leaves. In this Figure, in the tree on the left, 3 null hypotheses have changed from being rejected to being accepted compared to 2 in the tree on the right. In order to maximize the power of the procedure, we should choose to move the rejections up the tree in such a way as to minimize the number of null hypotheses that change from rejected to accepted. Thus we would prefer the tree on the right. In these figures the null hypotheses in red have been rejected and the null hypotheses in blue have been accepted.

We recall that N is the number of false rejections and R it the total number of rejections. For what follows, the definition below will be helpful:

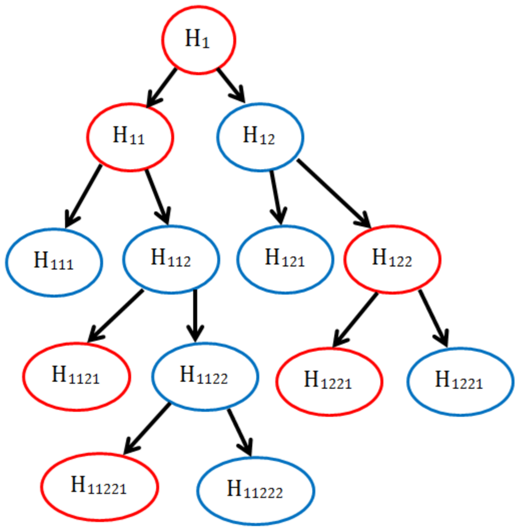
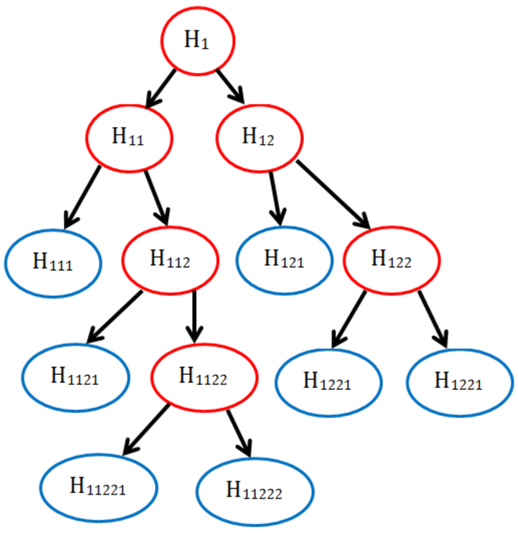
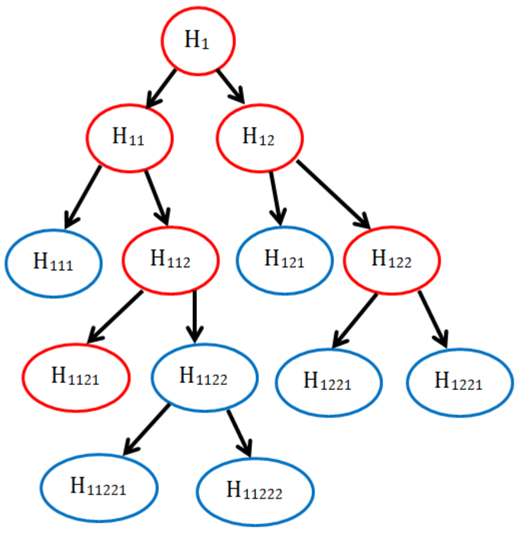


Figure 8: This is an example of a tree where a multiple hypothesis testing procedure has been applied. Red = Rejected, Blue = Accepted.

Definition 10. For a given rejection set $\mathcal{R}$ let $R(\mathcal{R}) = |\mathcal{R}|$ be the total number of rejections and let $N(\mathcal{R})$ be the number of false rejections, ie: the number of null hypotheses in $\mathcal{R}$ which are true.



(a) Application of *TreeShuffle*



(b) A different application of *TreeShuffle*

Figure 9: Here we have shown two different applications of *TreeShuffle* to the tree from Figure 8. Red = Rejected, Blue = Accepted.

Lemma 3. *Given any tree $\mathcal{T}$ and a corresponding rejection set $\mathcal{R}$, if $\mathcal{R}' =$ TreeShuffle$(\mathcal{T}, \mathcal{R})$ with $R' = R(\mathcal{R}')$ and $N' = N(\mathcal{R}')$, then we have: $R' = R$ and $N' \leq N$.*

Proof: The points at which $\mathcal{R}$ changes in *TreeShuffle* all occur at step 6 in Algorithm 7 from which it it clear that $|\mathcal{R}|$ is unchanged throughout so $R = R'$. Similarly the only point at which $N(\mathcal{R})$ changes is step 6. In the Algorithm, rejections always move up their branch, they never go down it. Given a branch: $B = (H_1, \ldots, H_n)$ we have that the first $k(B)$ hypotheses are false for some $k(B)$ and the rest are true, so moving rejections up the branch will always decrease the number of false rejections. This applies for each branch so it follows that $N' \leq N$. □

This is great because it means that we can apply our multiple testing procedure to the tree, apply TreeShuffle to the resulting rejection set and still control the number of false rejections. Since the *FWER* and *FDR* are functions of N and R (see section 3.2) we have the following theorem:

Theorem 7. *Suppose we have a tree: $\mathcal{T} = (H_1, \ldots, H_n)$ where each H_i has associated p-value: p_i and suppose that the p-values satisfy the conditions of a multiple hypothesis testing procedure: $Mhyp$. Let $\mathcal{R}$ be the rejection set which results after applying $Mhyp$ to the p-values. Then the FDR of TreeShuffle($\mathcal{T}$,Mhyp) is less than or equal to the FDR of Mhyp and the $FWER$ of TreeShuffle($\mathcal{T}$,Mhyp) is less than or equal to the $FWER$ of Mhyp.*

Proof: This follows from applying Lemma 3 and using the definitions of *FWER* and *FDR*

Algorithm 7 Shuffling the hypotheses in a tree after applying Mhyp

```
1: procedure TreeShuffle($\mathcal{T}, \mathcal{R}$)
2:     If $|\mathcal{T}| = |\mathcal{R}|$, then all hypotheses in $T$ are rejected so return $\mathcal{R}$
3:     for each leaf: $L$ in the tree $\mathcal{T}$ do
4:         if $L \notin R$ then set $\mathcal{T} \leftarrow \mathcal{T} \backslash L$
5:         else
6:             If there exists some $A \in \mathcal{A}(L)$ such that $A \notin R$, add $A$ to $\mathcal{R}$ and remove $L$
               from $\mathcal{R}$. If there is more than one such $A$ then choose the one highest up the tree.
7:             Set $\mathcal{T} \leftarrow \mathcal{T} \backslash L$.
8:         end if
9:     end for
10:    Repeat the above for loop until $|\mathcal{T}| = |\mathcal{R}|$. Then return $\mathcal{R}$.
11: end procedure
```

from Section 3.2. □

The great thing about this is that we can turn any multiple testing procedure into a hypothesis tree testing procedure just by applying it to the p-values and then applying *TreeShuffle*. Note also that by the same logic as above, because the acceptances are sent towards the bottom of the tree, the power of procedure (upon applying *TreeShuffle*) will increase (relative to *Mhyp*). This is because the number of false acceptances will decrease.

3.4.2 Applying *TreeShuffle*

As an example we take *BHtree* to be the procedure *TreeShuffle*($\mathcal{T}$, *BH*) where *BH* is the Benjamini-Hochberg procedure. Then *BHtree* will control the FDR to a level which is better than that controlled by Benjamini-Hochberg and under the same assumptions, namely that: $p_i, i \in I_0$ are independent, and independent of $\{p_i : i \notin I_0\}$. Indeed as discussed in Section 3.2, we can loosen these assumptions to that of positive dependence in the p-values.

Ideally it would be nice to prove that the p-values resulting from using Binary Segmentation are positively dependent, though I haven't looked into trying to do so. If we were unable to prove this or in the more general case where we had some arbitrary dependence in the p-values we could use Benjamini-Yekutieli as our multiple hypothesis testing procedure. Applying *TreeShuffle* would then control the *FDR* while rejecting null hypotheses in the right fashion. The issue with this is the same as the issue with the Benjamini-Yekutieli procedure. Namely it requires a large decrease in power in order to ensure that the *FDR* is still controlled.

3.4.3 Further Remarks

We have shown that applying the *TreeShuffle* procedure to *Mhyp* ensures that the *FDR* (or *FWER*) is controlled to the same level as *Mhyp*. However, most of the time we would expect the *FDR* to decrease, if we were able to work out a method to tell how much it decreased by, then we would be able to increase the power of our methods. This would work by increasing the control on the *FDR* of *Mhyp* while still being able to control the *FDR* of the tree of hypotheses to a given level. This would be an interesting next step.

Another thing to note is that given the tree if the mapping described in Section 3.3.4 doesn't result in much mixing of the alternative and null hypotheses, then we will (more or less) be in the False then true situation. Thus another approach to dealing with trees of hypotheses would be to first apply the mapping and then apply methods for the *False then True* problem. (Such as the ones we describe below in Chapter 4). The trouble is that this would not be guaranteed to control the *FDR*. However we could use it as a method and preliminary simulations indicate that it would work reasonably well. This would be especially relevant in the case where we had some measure of the amount of mixing between false and true hypotheses caused by the mapping. In this case one would be able to control the *FDR* using *False then True* methods.

Another potential means of applying methods for *False then True* to the hypothesis testing tree problem would be to note that by definition each branch is in *False then True* format. Thus it may be possible to apply *False then True* methods branch-wise.

4 False Then True

We will now develop Algorithms which look at the more specific problem of *False then True*. Recall this is the problem where we have null hypotheses: $H_1, H_2 \ldots, H_m$, where we know that the first k are false and the last $m-k$ are true for some unknown k and we must reject $H_1, H_2 \ldots, H_{\hat{k}}$ and accept $H_{\hat{k}+1}, H_{\hat{k}+2} \ldots, H_m$ for some $\hat{k} \in \{0, \ldots, m\}$.

Forward.Stop is a method tailored for dealing with the Sequential Hypothesis Testing problem. It works well however it has a tendency to underestimate k (see Section 4.2.4 for illustrations of this) and is not as powerful as it could be in the sense that it tends to accept many null hypotheses that are in fact false. We take advantage of the structure of the *False then True* problem in order to develop new techniques in Section 4.2 and show that they control the FDR while being much more powerful than $Forward.Stop$. However first we will consider an alternative means of dealing with the *False then True* problem that involves considering changepoints in distribution.

4.1 Changepoints in Distribution

Instead of describing a procedure which controls the FDR to some level α we look to identify the value of k (the point of the last false null hypothesis in the sequence). We will assume that we have a test statistic: K for evaluating each hypothesis pair and that K takes different distributions under the alternative and null hypotheses; this will be true for any sensible statistic in order to have some measure of significance. We can often assume that K has a continuous CDF and so is uniformly distributed under the null hypothesis, however in this section it will be sufficient to assume that the distribution is different.

We then can rather neatly consider the *False then True* problem as a changepoint problem, where here the changepoint is in the distribution of p-values. Assuming the change occurs at k, we expect that the p-values have different distributions either side of k. (Usually we will expect the p-values before k have a tendency to be significant and will tend to be lower than the p-values after k.) So we would like to apply methods that can identify when a change in distribution occurs. Since we are assuming we have no knowledge of the distribution of the p-values under the alternative hypothesis, (other than some rather vague idea that they should be smaller than uniform) we need to create a non-parametric method.

4.1.1 Non-Parametric Changepoint Detection

We have changed our hypothesis testing problem into that of a changepoint problem, and we need to consider means of identifying the changepoint. For now let us consider a more general setting and assume the following model:

Model 2. *We have observations: $X_1, X_2, \ldots, X_n$ (for some $n \in \mathbb{N}$), and there is some $k \in \mathbb{N} \cup \{0\}$ such that $X_1, \ldots, X_k$ are i.i.d from some distribution with CDF: F_L and $X_{k+1}, \ldots, X_n$ are i.i.d from some distribution with CDF F_R.*

Our problem is then to identify k. What we would like to do intuitively is to find the point at which the difference between the distribution of the points on the left of the potential changepoint and the points on the right of it is maximized. (Much like when we considered changepoints in the mean.) In order to go about this, it will be helpful for us to have some means of distinguishing between distributions. For this purpose we shall now consider the Kolmogorov-Smirnov distance between two CDFs. This is defined by:

Definition 11. Given two cumulative distribution functions: F, G, we define the **Kolmogorov-Smirnov distance** between them to be: $\mathbf{KS(F,G)} = \sup_{x\in\mathbb{R}}|F(x) - G(x)|$

The following definition will also prove helpful:

Definition 12. For $X_1, \ldots, X_n$ i.i.d, we define the **estimated CDF** (ECDF) to be:
$\hat{F}(x) = \frac{1}{n}\sum_{i=1}^{n} 1\left[X_i \leq x\right]$ and for $a < b$, define $\hat{F}_{a,b}(x) = \frac{1}{b-a+1}\sum_{i=a}^{b} 1\left[X_i \leq x\right]$.

We have the Glivenko Cantelli Theorem, see van der Vaart, A.W. and Wellner (1996):

Theorem 8. *If $X_1, \ldots, X_n \overset{iid}{\sim} F$, then $KS\Big(\hat{F}(X_1, \ldots, X_n), F\Big) \to 0$ as $n \to \infty$*

Now given a sample $X_1, \ldots, X_n$ following Model 2, for each potential changepoint $j = 1, \ldots, n$, we want to compute a value $\Delta(j)$ which is a measure of the difference in distribution between the sample to the left and to the right of j. We will use Kolmogorov Smirnov as our means of measuring this difference. Note that there are other means for distinguishing between distributions, such as the Cramer Von Mises statistic. We will be using the Kolmogorow Smirnov distance, so we denote Δ by Δ_{KS}, which for $j = 1, \ldots, n-1$ we take to be:

$$\Delta_{KS}(j) := KS\left(\hat{F}_{1,j}(x), \hat{F}_{j+1,n}(x)\right) = max_{i=1,\ldots,n}\left|\hat{F}_{1,j}(X_i) - \hat{F}_{j+1,n}(X_i)\right| \tag{1}$$

The second equality in (1) holds because the value of the Kolmogorov Smirnov distance between the two estimated CDFs only changes at the points X_i. This formula makes the difference easier to compute as we don't have to evaluate it for each point x.

Having calculated the value of the Δ_{KS} at each point, we would like to use these values to calculate the location of the changepoint. One way of doing this would be to choose the j such that $\Delta_{KS}(j)$ is maximal, however this runs into some problems. To illustrate this we simulate a series of non-parametric changepoint problems and plot the values of Δ_{KS} at each point. See Figures 10 and 11. In these figures, $n = 200$ and the true value of k is 100.

From these simulations, we see that despite the fact that the change occurs at $k = 100$, Δ_{KS} is not necessarily maximized at this point. This is because there is no account taken for the effect of sample size. If for instance the first point sampled is not representative of its distribution, that will lead to a large difference between $\hat{F}_{1,1}(x)$ and $\hat{F}_{2,n}(x)$ and thus a large value of $\Delta_{KS}(1)$, despite the fact that the changepoint does not occur there. There are existing methods to deal with this effect, see Ross and Adams (2012).

Here we consider an alternative method which still uses the Δ_{KS} statistic, however doesn't seek to maximize it. Instead it uses the fact that after k we expect the value of Δ_{KS} to decrease. This is because (for $j > k$), $\hat{F}_{1,j}$ includes $j - k$ points from F_R and k points from F_L, so once we have $j > k$, increasing j will cause $\hat{F}_{1,j}$ to get closer and closer to F_R.Of course this will not necessarily true at every point (just by randomness) however we would expect to have $\Delta_{KS}(j) > \Delta_{KS}(j+1)$ for most $j > k$ and to have $\Delta_{KS}(j) > \Delta_{KS}(j+1)$ for considerably fewer $j > k$. (Similarly we have the opposite to the left of the changepoint: we expect Δ_{KS} to usually be increasing.) Thus to identify k we can look to see where $\Delta_{KS}(j)$ starts to change from increasing to decreasing. Examination of the plots in Figures 10 and 11 below reveal that this gives very good estimate of k (the true value in each graph being $k = 100$) as there is a clear peak in each case at $t = 100$.

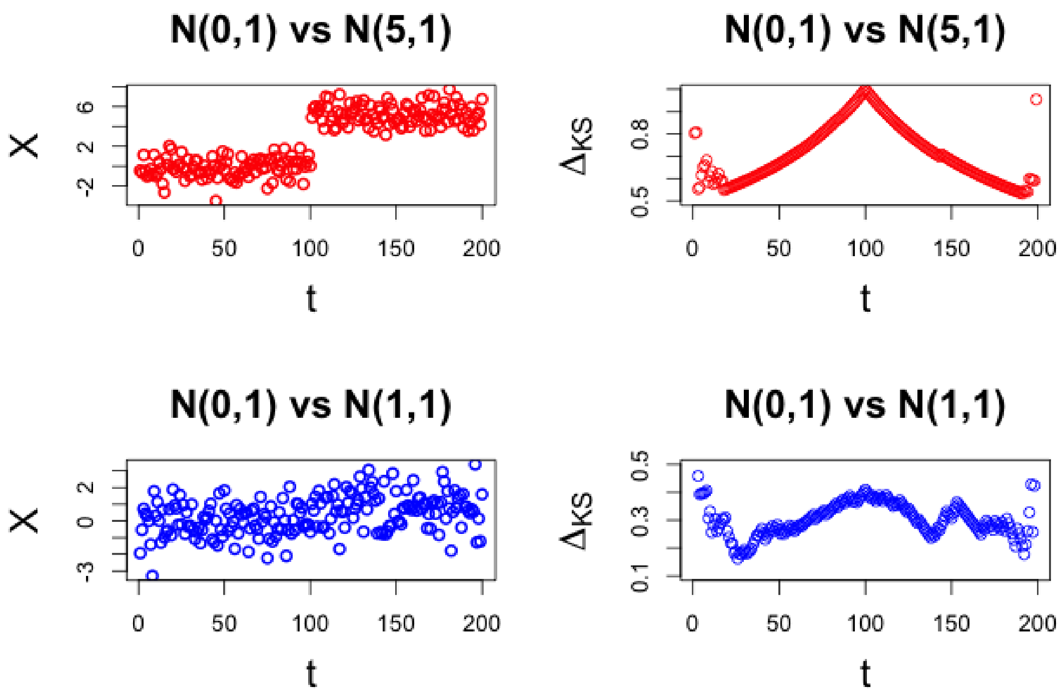


Figure 10: (From Section 4.1.1) In the top left we have data $X_t \sim N(0,1)$ for $1 \le t \le 100$ and $X_t \sim N(5,1)$ for $101 \le t \le 200$ and the Δ_{KS} statistic calculated in at each point in the top right. (Similarly in the bottom panels.)

The question is then how to measure when change occurs. This procedure may slightly

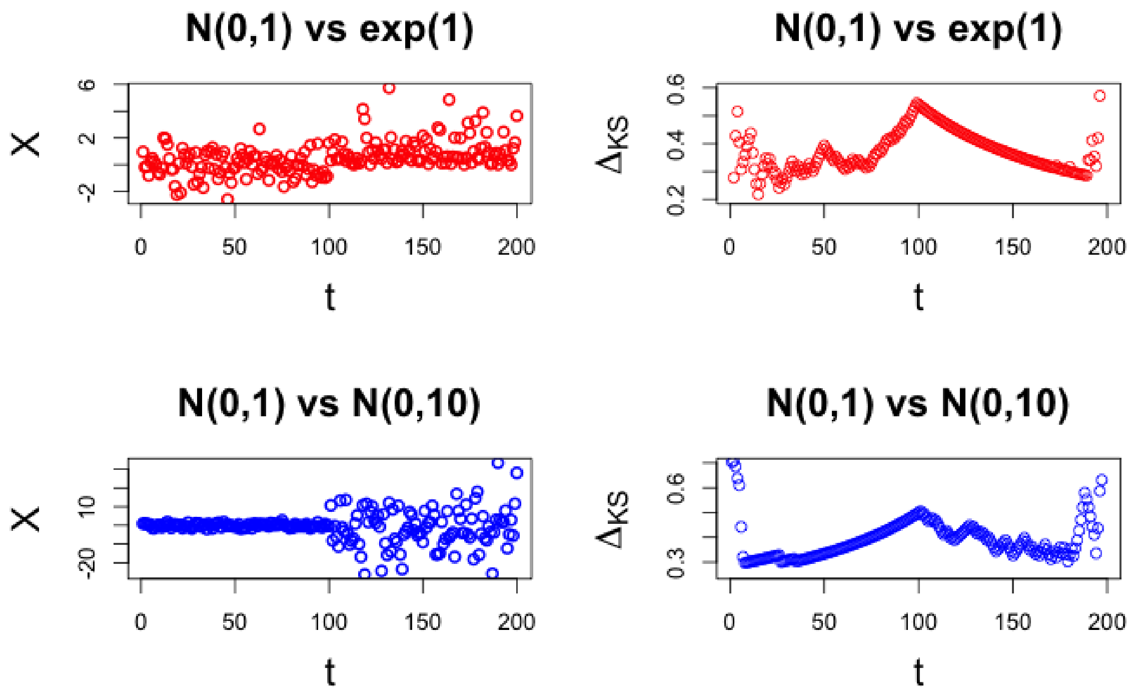


Figure 11: (From Section 4.1.1) In the top left we have data $X_t \sim N(0,1)$ for $1 \leq t \leq 100$ and $X_t \sim exp(1)$ for $101 \leq t \leq 200$ and the Δ_{KS} statistic calculated in at each point in the top right. (Similarly in the bottom panels.)

over or underestimate k, but the more data points that are available either side of k, the better it will be at identifying the change. Of course, if we know that the true value of the changepoint does not occur at the first or last few points of the interval, then this procedure works even better as we eliminate any potential small sample bias.

4.1.2 Detecting a Change in Slope

Motivated by the above discussion, we can consider the following change in slope model:

Model 3. $X_t = a_t + b_t t + \epsilon_t$ *where* $a_t = \begin{cases} a_1 & t \leq k \\ a_2 & t > k \end{cases}$ *and* $b_t = \begin{cases} b_1 & t \leq k \\ b_2 & t > k \end{cases}$

for some constants: $a_1, a_2, b_1, b_2 \in \mathbb{R}$ and some $k \in \{1, \ldots, n-1\}$ and where ϵ_t is some noise term. Now let $Y_t = X_{t+1} - X_t$ for $1 \leq t \leq n-1$.

We have that for $t < k$, $Y_t = b_1 + \epsilon_{t+1} - \epsilon_t$ and for $t \geq k+1, Y_t = b_2 + \epsilon_{t+1} - \epsilon_t$. So taking $\zeta_t = \epsilon_{t+1} - \epsilon_t$, we have that Y_t follows Model 1 that we introduced in Section 2 (except at

$t = k$), indeed:

$$Y_t = f_t + \zeta_t \text{ where } f_t = \begin{cases} b_1 & t < k \\ b_2 & t > k \end{cases}$$

So we can apply the techniques for identifying changepoints from Section 2 to our change in slope problem. We will call this method Slope Changepoint Detection (or *SCD* short). *SCD* could use any method to identify the change in Y however we shall use the CUSUM statistic as in Section 2.1.1. Supposing that $X_1, \ldots, X_n$ follow Model 3 we generate an estimate of k: $\hat{k}$ via Algorithm 8 below.

See Figure 12 for an example of this in action. In this example we have generated data in the form of Model 3 where the $\epsilon_t \overset{iid}{\sim} N(0, 5)$ and calculated the difference sequence Y. Applying the SCD algorithm we get an estimated changepoint of 100, (which is the true location). Note that for this example we have chosen to make the underlying data continuous, however this need not be the case for the method to apply. Note also that here we are assuming that a change does occur so we do not need to include hypothesis testing.

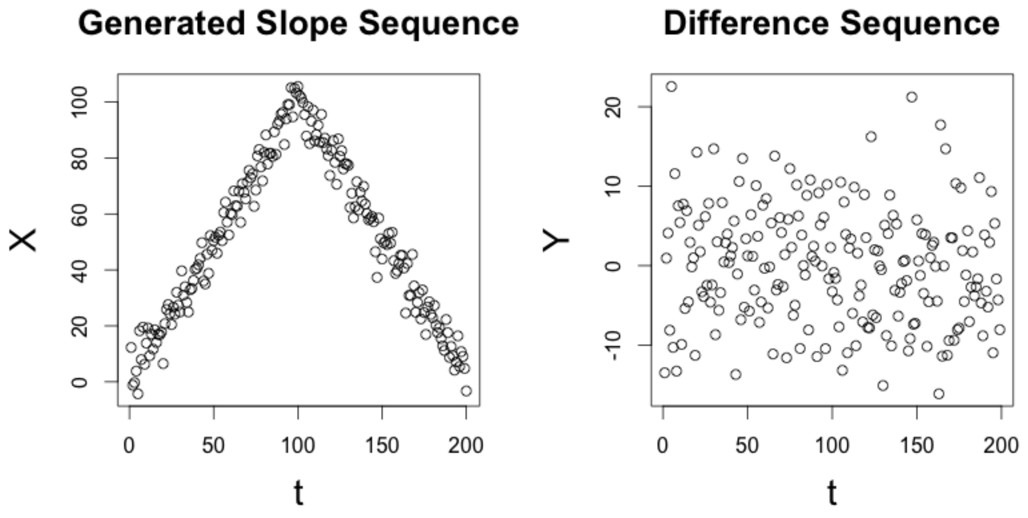


Figure 12: *SCD* in action.

Now we describe an Algorithm that uses this and the reasoning at the end of Section 4.1.1 to identify a changepoint in distribution. Suppose we have data: $X_1, \ldots, X_n$ following Model: 2, then our estimate for the true value of the changepoint: k is generated by Algorithm 9 (see below).

4.1.3 Sign Changes

Unfortunately, it turns out that Algorithm 9 is not very good at distinguishing the variation at the side of the Δ_{KS} statistic from the main feature that we would like to identify: the

Algorithm 8 Slope Changepoint Detection

```
1: procedure SCD($X_1, \dots, X_n$)
2:     for $t = 1, \dots, n-1$
3:         $Y_t = X_{t+1} - X_t$
4:     end for
5:     for $j = 1, \dots, n-1$
6:         Compute: $\tilde{Y}^j = \left| \sqrt{\frac{n-j-1}{(n-1)j}} \sum_{t=1}^{j} Y_t - \sqrt{\frac{j}{(n-1)(n-j-1)}} \sum_{t=j+1}^{n-1} Y_t \right|$
7:     end for
8:     $\hat{k} = argmax_{\{j=1,\dots,n-1\}}\{\tilde{Y}^j\} + 1$    ▷ The plus 1 is to account for the changing index.
9: end procedure
```

Algorithm 9 Changepoint in Distribution via Δ_{KS} and SCD

```
1: procedure KS-SCD($X_1, \dots, X_n$)
2:     for $t = 1, \dots, n-1$
3:         $Z_t = \Delta_{KS}(t)$
4:     end for
5:     $\hat{k} = SCD(Z_1, \dots, Z_{n-1})$
6: end procedure
```

change in slope. See Figure 13 below; notice that the CUSUM graph doesn't work well at all, it is certainly not maximized at the changepoint. In fact in both of these examples it is actually minimized there. The changepoint is thus (by this method) estimated to be 1 in the first case and 193 in the second case where the true value is 100!

In order to improve this, we can note that as we described earlier Δ_{KS} changes from generally increasing to generally decreasing. We can thus consider the sign of $\Delta_{KS}(j+1) - \Delta_{KS}(j)$ at each point as this will give us an idea of when the Δ_{KS} is increasing or decreasing. This motivates the SSC algorithm (for considering a change in sign) and the KSS algorithm for estimating the change in distribution. (see Algorithms 10 and 11 below). It turns out these work much better.

To see how this *KSS* works see Figure 14 where we have shown the sign sequence that results when applying the *SSC* algorithm to the examples from Figures 10 and 11. Applying the *KSS* algorithm (by maximizing the CUSUM statistic applied to the plots in Figure 14) gives changepoint point estimates of 100, 100, 99 and 101. This shows that the *KSS* is working well, since the true changepoint is at 100.

I haven't looked into proving it here, but I think that it would be possible to prove consistency results about KSS (especially because of its reliance on the CUSUM statistic for the second step, and the fact that there already exist consistency results about the CUSUM statistic, see Fryzlewicz (2014)). In the above, using the Kolmogorov-Smirnov statistic we

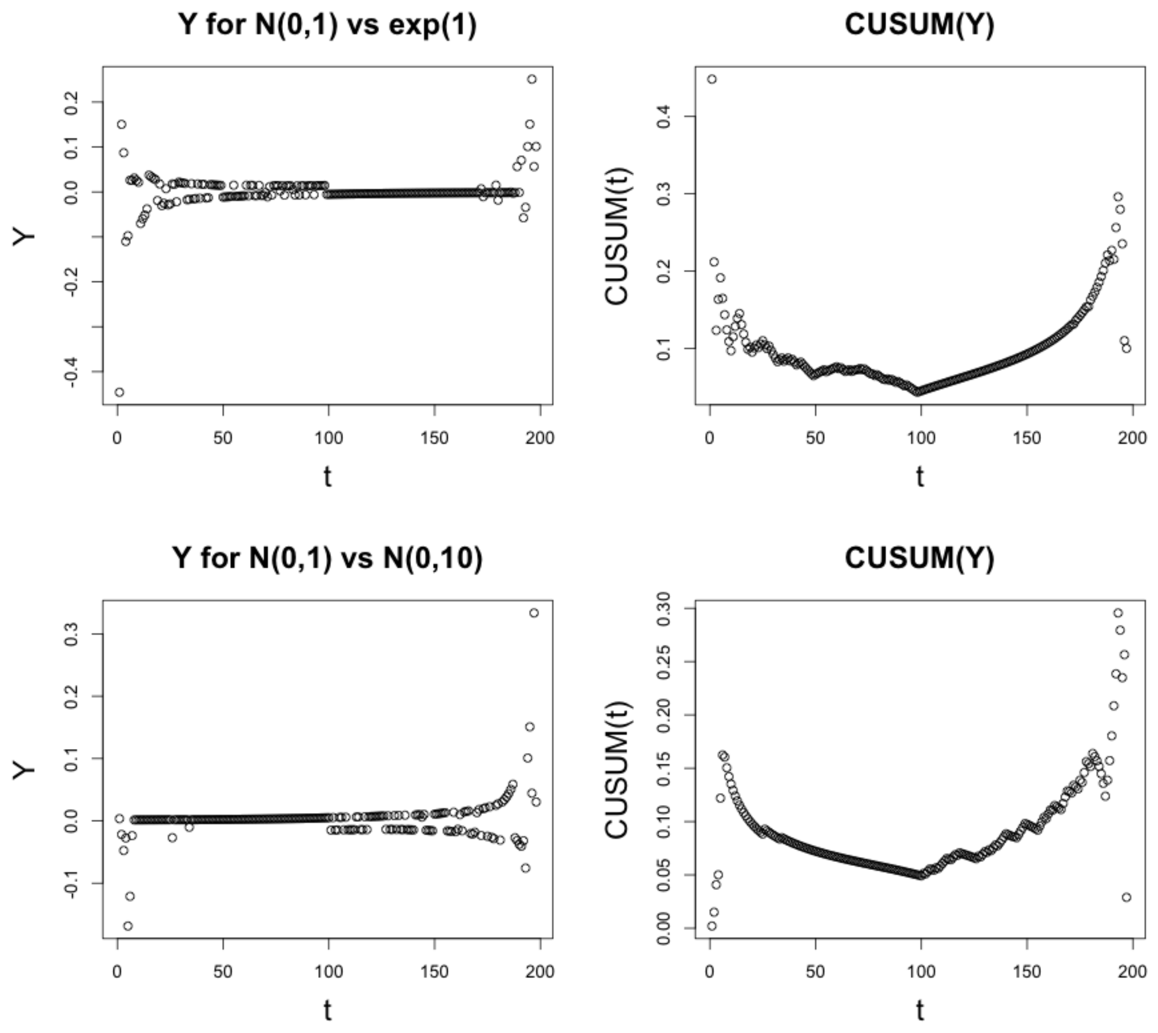


Figure 13: (From Section 4.1.3) Application of KS-SCD. In the top right panel we see the difference sequence Y for the $N(0,1)$ vs $exp(1)$ graph from Figure 11 and the CUSUM statistic applied to this in the top right panel. Similarly in the bottom panels but for the $N(0,1)$ vs $N(0,10)$ graph from Figure 11.

have been assuming the data is independent. This is required if the estimates of F_L and F_R are to converge at the left and the right. However, even if there is some weak dependence among the data, so long as there is a change in distribution this method will be fairly robust and still work well. As such we would still recommend the use of this method in such situations. One thing to note is that it may be possible to develop methods which are specifically tailored to allow some sort of dependence in the data (and then use these in place of *KSS* to identify k), I have not looked into this in detail.

It should be noted that *KSS* performs rather well as a changepoint detection method; preliminary simulations suggest that it performs almost as well as the CUSUM-statistic in

Algorithm 10 Slope Sign Change

```
1: procedure SSC($X_1, \ldots, X_n$)
2:     for $t = 1, \ldots, n-1$
3:         $Y_t = X_{t+1} - X_t$
4:     end for
5:     $Z \leftarrow (Y > 0)$                    ▷ A vector with entries: $Z_i$=1$[Y_i > 0]$
6:     for $j = 1, \ldots, n-1$
7:         Compute: $\tilde{Z}^j = \left|\sqrt{\frac{n-j-1}{(n-1)j}} \sum_{t=1}^{j} Z_t - \sqrt{\frac{j}{(n-1)(n-j-1)}} \sum_{t=j+1}^{n-1} Z_t\right|$
8:     end for
9:     $\hat{k} = argmax_{\{j=1,\ldots,n-1\}}\{\tilde{Z}^j\} + 1$     ▷ The plus 1 is to account for the changing index.
10: end procedure
```

Algorithm 11 Kolmogorov - Smirnov Change in Sign

```
1: procedure KSS($X_1, \ldots, X_n$)
2:     for $t = 1, \ldots, n-1$
3:         $\delta_t = \Delta_{KS}(t)$
4:     end for
5:     $\hat{k} = SSC(\delta_1, \ldots, \delta_{n-1})$
6: end procedure
```

the mean change case. This is rather good given that *KSS* is designed to consider a general changepoint in distribution, rather than being specific to the mean change case.

4.1.4 Changepoints in the p-values

Having discussed methods for identifying changepoint in distribution, we now look to apply the above techniques to our False then True problem. For any sensible choice of test-statistic the distribution of the p-values will be different under the null and alternative hypotheses and so we can treat our problem of identifying k as a problem in identifying a changepoint in distribution and our estimate of the changepoint will be our estimate of k.

Given this, we propose the following method for identifying the value of k in the False then True scenario (as described in Section 3.3.1):

Changepoint.Stop takes $\hat{k} = KSS(p_1, \ldots, p_T)$

See Section 4.2.4 for simulations of *Changepoint.Stop* applied to sequences of p-values, it is shown to do very well. Note that because of this format we will often refer to k in the False then True problem as the **changepoint**, even when we are not using changepoint methods eg in Section 4.2.

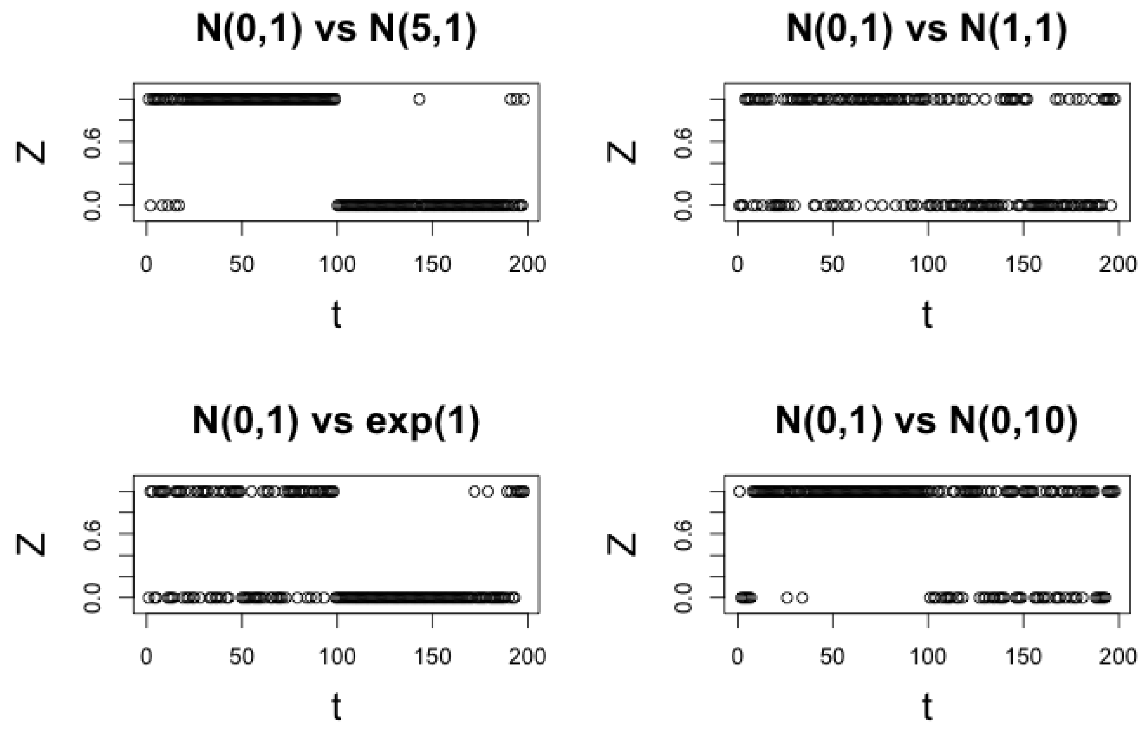


Figure 14: Application of KSS. Here we have plotted the sign sequence: Z for each of the examples.

Note that we do not need the p-values to be independent in order for this method to apply. So long as there is a change in distribution, even under weak dependence *KSS* will still work well.

Assuming the test-statistic has a continuous CDF, we have that the p-values are uniformly distributed under the null hypothesis. We could probably incorporate this fact to improve the Algorithm, however we have not yet done this here. As such this algorithm applies more generally to test-statistics whose CDFs are not necessarily continuous.

4.1.5 Further Remarks

We note that we haven't proved results regarding control of the *FDR* for this method I think it's definitely possible (though perhaps difficult) to do so, especially if we were to make a modification such as that of Decrease.Stop (see Section 4.2.3).

Another observation is that we could consider using online methods for identifying the changepoint in distribution. (This is when one attempts to estimate the change as soon as possible after it occurs.) These are useful because they can be used in order to reduce the amount of computation required (for instance in application to trees where calculation of lots of bootstraps at each stage can be computationally intensive). See Kifer et al (2004) for an example of an online method for detecting a changepoint in distribution with independent data. It should be noted that the methods developed in the next section are online methods.

Something else to note is that the methods described above for detecting a changepoint in distribution can of course be generalized even further to situations where there are multiple changepoints, where we could consider applying some sort of Binary Segmentation algorithm. For our purposes, motivated by the *False then True* Problem, we have only considered the case where there is one changepoint. However it would be interesting to examine this extension. When considering the identification of multiple changepoints in distribution and determining whether one exists we could introduce hypothesis testing at each stage. Obviously this is not sensible for the p-value situation because we know that there is a changepoint at some point (though it could be at 0 in the case where all hypotheses were null), but could be considered as a general method for detecting changes in distribution.

4.2 Algorithms for False then True

We have described a means of tackling *False then True* using a changepoint approach above, however while this works well in practise it is difficult to prove results about it. In fact we can take advantage of the structure of the *False then True* problem to invent methods which have improved power (relative to *Forward.Stop*) while still controlling the *FDR*. This is possible because if you look at the tables in Section 4.2.4 we see that in practice *Forward.Stop*(α) method actually has an FDR which is much lower than α even though it has been controlled to a level α. This means that there is substantial room for improvement.

When controlling the *FDR*, *Forward.Stop* requires that all of the p-values are independent. For the methods in this Section we will only require the assumption that:

Assumption 1. *The p-values: p_i, $i \in I_0$ are independent, and independent of $\{p_i : i \notin I_0\}$.*

Note that this is the same assumption that the Benjamini-Hochberg procedure requires in the general multiple hypothesis testing setting.

4.2.1 Uniform.Stop

The following simple approach which applies to the general Sequential Hypothesis Testing scenario as well as to the *False then True* problem, is described in Marcus et al (1976).

Uniform.Stop(α) takes $\mathcal{A} = \{i \in \{0, \ldots, m-1\} : p_{i+1} > \alpha\}$ and sets $\hat{k} = \min\{\mathcal{A}\}$ if $\mathcal{A} \neq \emptyset$ and set $\hat{k} = m$ otherwise.

Theorem 9. *Uniform.Stop(α) controls the FWER (and therefore the FDR) to a level α.*

Proof: $\{N \geq 1\} = \{\hat{k} > k\}$. So $FWER = \mathbb{P}(N \geq 1) = \mathbb{P}(\hat{k} > k) \leq \mathbb{P}(p_{k+1} \leq \alpha) \leq \alpha$. By Theorem 5, $FDR \leq FWER$ so this procedure controls the FDR to a level α as well. □

As discussed in G'Sell et al (2015), an issue with this method is that, even when there are very small p-values, it will underestimate the location of k if these small values are preceded by a few medium-sized p-values. This can cause it to have low power. Note that *Uniform.Stop* controls the $FWER$ in the general Sequential Hypothesis Testing scenario, not just the False then True case (the proof is an easy extension of the above).

4.2.2 Normal.Stop and Extended.Stop

We propose the following procedures in order to solve the problems with power:

$$\textbf{Normal.Stop}(q, n) \text{ takes } \mathcal{B} = \{i \in \{0, \dots, m-1\} : p_{i+j} > q \text{ for } j \in \{1, \dots, n\}\}$$
$$\hat{k} = \max\{\min\{\mathcal{B}\}, 0\} \text{ if } \mathcal{B} \neq \emptyset \text{ and sets } \hat{k} = m \text{ otherwise.}$$

$$\textbf{Extended.Stop}(q, n, C) \text{ takes } \mathcal{B} = \{i \in \{0, \dots, m-1\} : p_{i+j} > q \text{ for } j \in \{1, \dots, n\}\}$$
$$\hat{k} = \max\{\min\{\mathcal{B}\} - C, 0\} \text{ if } \mathcal{B} \neq \emptyset \text{ and sets } \hat{k} = m \text{ otherwise.}$$

Where C, q, n are constants such that $C, n \in \mathbb{N}$ and $q \in [0, 1]$. These are chosen to ensure that the procedure controls the $FWER$ (or FDR if desired) to a level α. Note that the outer maximum above is introduced to ensure that $\hat{k} \geq 0$. We will see that these procedures are more powerful than *Forward.Stop* in the *False then True* case and that we can still ensure that they control the *FWER/FDR*. Note that *Normal.Stop*(q, n) = *Extended.Stop*$(q, n, 0)$. Hereon we will mainly refer to *Extended.Stop* since it is a generalization of *Normal.Stop*.

Intuitively, what *Normal.Stop* does is to look for the start of the first string of n heads (where heads here corresponds to observing a p-value which is larger than q) and uses the fact that to the right of k we know that the p-values are uniformly distributed and independent. To control the FDR we can use the probability that the first string of n heads occurs at a given point (see Theorem 10). The first string may occur to the left of k (if our p-values are not behaving there) in which case we will have underestimated the true value of k and will have no false rejections, or it may occur after k. We can calculate the probability of it occurring after k and can choose our parameters such that choosing $\hat{k}$ as our estimate results in a low chance of false rejections.

Definition 13. For the following proofs and discussion it will be helpful to define $z = \max\{\min\{\mathcal{B}\}, 0\} + 1$ if $\mathcal{B} \neq \emptyset$ and set $z = m$ otherwise. (Where $\mathcal{B}$ is as above.) In other words we take z to be the point at which the first string of n heads occurs.

In order to discuss parameter choice and controlling the *FWER/FDR*, we need to calculate what the error probabilities are going to be. For this it is helpful to consider the following:

Theorem 10. *Consider flipping a coin with probability of tails: q (where each flip is independent of the others), let $A_{i,n}$ be the event that the first string of n heads starts at the ith flip and let $a_{i,n} = \mathbb{P}(A_{i,n})$. Then $a_{1,n} = (1-q)^n$ and $a_{i,n} = \left(1 - \sum_{j=1}^{i-n-1} a_{j,n}\right) q(1-q)^n$ for all $i > 1$.*

Proof: $a_1 = (1-q)^n$ as the flips are independent. Let $t_j \in \{H, T\}$ be the value that the coin takes on the j toss. Now, $\mathbb{P}(A_{i,n}) = (1-q)\mathbb{P}(A_i \mid t_{i-1} = H) + q\mathbb{P}(A_{i,n} \mid t_{i-1} = T) = q\mathbb{P}(A_{i,n} \mid t_{i-1} = T)$ as if $t_{i-1} = H$, then the first string of n heads cannot start at i, so $\mathbb{P}(A_{i,n} \mid t_{i-1} = H) = 0$.)

Now if $t_{i-1} = T$, then there cannot be a string of n heads starting at any $i-n \le j \le i-1$(as it is physically impossible), so $\mathbb{P}(A_{i,n} \mid t_{i-1} = T) =$
$\mathbb{P}$(A string of n heads starts at i and no string of n heads starts at any $j < i \mid t_{i-1} = H$) $=$
$\mathbb{P}$(A string of n heads starts at i and no string of n heads starts at any $j < i-n | t_{i-1} = H$) $=$
$\mathbb{P}$(A string of n heads starts at i and no string of n heads starts at any $j < i-n$) $=$
$\mathbb{P}$(A string of n heads starts at i)$\mathbb{P}$(no string of n heads starts at any $j < i-n$)
The last two equalities hold because of the independence of the tosses.

So $\mathbb{P}(A_{i,n} \mid t_{i-1} = T) = (1-q)^n \left(1 - \sum_{j=1}^{i-n-1} a_{j,n}\right)$.
Thus it follows that $\mathbb{P}(A_{i,n}) = q\mathbb{P}(A_i | t_{i-1} = T) = \left(1 - \sum_{j=1}^{i-n-1} a_{j,n}\right) q(1-q)^n$ as required. □

Now it is important to note that the above values are not very tractable. However we can easily calculate them using a computer.

We can apply the above Theorem to our False then True problem. To do this, we will take advantage of the ordering of the hypotheses and notice that the right of the changepoint, the probability of observing the first n such heads in a row at a given point can be calculated and thereby used to calculate and to control the *FDR*. (Recall we refer to the true value of k in the False then True problem as the **changepoint** because of the discussions in Section 4.1.)

The constant C gives us extra control over the procedure. Increasing the value of C will decrease the *FDR* because it will reduce $\hat{k}$. Also, increasing the value of n or decreasing the value of q will increase the power, because the point at which the first n heads are observed will become more likely to be observed sooner.

Theorem 11. *Extended.Stop(q, n, C) controls the FWER (and therefore the FDR) to a level: $1 - \sum_{i=1}^{C+1} a_{i,n}$ where $a_{i,n} = \left(1 - \sum_{j=1}^{i-n-1} a_{j,n}\right) q(1-q)^n$ from Theorem 10.*

Proof: $FWER = \mathbb{P}(N \ge 1) = \mathbb{P}(\hat{k} > k)$. Now we have that $\hat{k} = z - 1 - C$, where z is as defined in definition 13, thus it follows that:

$$\{\hat{k} > k\} = \{z > k + C + 1\} = \bigcup_{i=C+2}^{m} \{z = i + k\}$$

$$\text{so } \mathbb{P}(\hat{k} > k) \leq \sum_{i=C+2}^{m} \mathbb{P}\left\{z = i + k\right\} \text{ (as the events in the union are disjoint.)}$$

Now, $\mathbb{P}\left\{z = i + k\right\} \leq a_{i,n}$ (as the p-values are iid $U[0,1]$ after k), so it follows that:

$$\mathbb{P}(\hat{k} > k) \leq \sum_{i=C+2}^{m} a_{i,n} \leq 1 - \sum_{i=1}^{C+1} a_{i,n} \text{ as } \sum_{i=1}^{\infty} a_{i,n} = 1 \text{ and so: } FWER \leq 1 - \sum_{i=1}^{C+1} a_{i,n} \quad \square$$

A corollary of this is that:

Theorem 12. *Normal.Stop(q, n) controls the $FWER$ (and therefore the FDR) to a level: $1 - (1-q)^n$. (As $a_{1,n} = (1-q)^n$ from Theorem 10).*

To give an example of how we could use this in practise, let us consider the case where $q = 0.2$, $n = 3$. Using R, and the results of Theorem 10, we calculate the first 10 values of the a_i and find that: $a_1 = 0.5120, a_2 = 0.1024, a_3 = 0.1024, a_4 = 0.1024, a_5 \approx 0.0500, a_6 \approx 0.0395, a_7 \approx 0.0290, a_8 \approx 0.0185, a_9 \approx 0.01340, a_{10} \approx 0.0094$. Then if we take $C = 3$ then under this procedure, $FWER \leq 1 - \sum_{i=1}^{C+1} a_{i,n} = 0.1808$. Note that the choice of C, q, n above was arbitrary and we could decrease the bound on the *FWER* to any desired level, by changing their values. To give another example, *Extended.Stop*(0.1,4,3) (using the same logic as above) controls the *FWER* to a level: 0.0523.

Theorem 13. *Extended.Stop(q, n, C) controls the FDR to a level: $\sum_{i=1}^{m-C-1} ia_{i+C+1,n}/(k+i)$ where $a_{i,n} = \left(1 - \sum_{j=1}^{i-n-1} a_{j,n}\right) q(1-q)^n$ from Theorem 10.*

Proof: For $n \in \mathbb{N}$, $\mathbb{P}(N = i) = \mathbb{P}(\hat{k} = k + i) = \mathbb{P}(z = k + C + 1 + i) \leq a_{C+1+i,n}$. So:

$$FDR = \mathbb{E}\left(\frac{N}{\max(R,1)}\right) = \sum_{i=1}^{m-k-C} \mathbb{E}\left[\frac{N}{\max(R,1)}\middle| N = i\right] \mathbb{P}(N = i)$$

$$\leq \sum_{i=1}^{m-k-C} \frac{ia_{i+C+1,n}}{k+i}$$

The equality holds as $N \leq m - k - C$ and the inequality holds as
$N = i$ implies that $\max(R, 1) = k + i$

$\square$

A corollary of this is that:

Theorem 14. *Normal.Stop(q, n) controls the FDR to a level: $\sum_{i=1}^{m-1} ia_{i+1,n}/(k+i)$ where $a_{i,n} = \left(1 - \sum_{j=1}^{i-n-1} a_{j,n}\right) q(1-q)^n$ from Theorem 10.*

If we wanted to control the *FDR* of these procedures without having to vary the value of q, we could consider the following procedure:

Randomized Extended.Stop(q, n, C, r) takes
$\mathcal{B} = \{i \in \{0, \ldots, m-1\} : p_{i+j} > q \text{ for } j \in \{1, \ldots, n\}\}$ and generates: $U \sim U[0,1]$ and sets:

$$\hat{k} = \begin{cases} \max\{\min\{\mathcal{B}\} - C, 0\} & \text{if } \mathcal{A} \neq \emptyset \text{ and } U \leq r \\ \max\{\min\{\mathcal{B}\} - C + 1, 0\} & \text{if } \mathcal{A} \neq \emptyset \text{ and } U > r \\ m & \text{otherwise} \end{cases}$$

With this procedure the *FDR* can be controlled by changing the value of r and keeping q constant.

Note that we have not sought to optimize the choice of the parameters in our examples. In order to do so we recommend taking account of the "round about" level of the initial p-values, in order to lead to a procedure which has the optimal balance of power and size. Further work could look into the choice of these parameters in greater depth.

4.2.3 Decrease.Stop

Partially inspired by the ideas behind $Normal.Stop$, we can consider doing something similar when it comes to considering identifying a changepoint in distribution (and hence in the p-values). This may not be as accurate as identifying the changepoint using *KSS*. However we think it will be easier to prove results about how it controls the FDR, though we have not looked into this. The method is:

Decrease.Stop(n) takes $\hat{k} = \min\{i : \Delta_{KS}(i) > \Delta_{KS}(i+1) > \ldots > \Delta_{KS}(i+n)\}$

This takes advantage of the fact that to the left of the changepoint we expect $\Delta_{KS}(j)$ to tend to be increasing and for it to be decreasing to the right of the changepoint. So we look for the first instance of a string of decreases.

The relation with $Normal.Stop$ is that, to the right of the changepoint, we will have $\Delta_{KS}(j) \leq \Delta_{KS}(j+1)$ some of the time and $\Delta_{KS}(j) > \Delta_{KS}(j+1)$ the rest of the time. Then if we regard the latter to be happening with a certain probability: q, and the former with probability $1-q$, then we are effectively looking for a string of heads just as in $Normal.Stop$. Here q corresponds to the probability that the Δ_{KS} statistic will decrease upon shifting the jth data point from the set of data on the left to the set of data on the right. This will not in fact be constant, which is where the analogy breaks down. This probability will be changing and this is what makes it difficult to prove that this method controls the *FDR* to a given level.

4.2.4 Method Performance

In order to demonstrate that the new methods we have proposed work better than *Forward.Stop* in the *False then True* context, we will now look at how well each of them works.

In order to compare how these procedures perform, we simulate p-values. In order to do so we will consider the scenario where $k = 20$ and $m = 100$. Here we have 100 null hypotheses which we would like to test where the first 20 are false and the remainder are true. This is an example of a situation we would see under the *False then True* set up. To simulate p-values we will generate $p_i \overset{iid}{\sim} Beta(1, B)$ for $1 \leq i \leq 20$ and $p_i \overset{iid}{\sim} U[0, 1]$ for $20 < i \leq 100$. In our examples we will take $B = 8, 14$ and 23. This is similar to the choice of simulation used in G'Sell et al (2015) though they also mix the null and non-null hypotheses. We will refer to the case $B = 8$ as the hard setting, $B = 14$ as the medium setting and $B = 23$ as the easy setting. The intuition behind this labelling is that the lower the value of B, the less significant the p-values to the left of the changepoint will be (because of how the Beta distribution works). So lowering the value of B makes it harder to correctly identify the location of the change and thus which hypotheses are false and which are true.

This is a natural way to test our procedure since in each of these cases the distribution of the p-values simulated from the Beta distribution (for these choices of B) will be skewed towards 0. This is what we want as for a good choice of test-statistic we would expect the distribution of the p-values to have some skew towards 0 under the alternative hypothesis. In the *False then True* scenario the new procedures that we developed have much higher power and still controlling the *FDR*. We see this in our simulations below. In order to compare them via simulation, we need to introduce a measure of the power in this context. What we will use is that of Average Power which is the same as that used in G'Sell et all (2015). If we apply a procedure to the p-values to yield an estimate $\hat{k}$ of k, this is defined as:

$$\textbf{Average Power (AP)} := \frac{\hat{k} - N}{k} = \begin{cases} \hat{k}/k & \text{if } \hat{k} \leq k \\ 1 & \text{if } \hat{k} > k \end{cases}$$

In the above definition N is the number of false rejections.

In order to test our procedures we can apply them to each set of p-values in turn. For each given procedure $\mathcal{P}$ we generate p-values as described above (independently of other simulations) and apply $\mathcal{P}$ to generate an estimate of k. We do this m times, for the jth such application of $\mathcal{P}$, $j = 1, \dots, m$, let $\hat{k}_j$ be the generated estimate of the changepoint, let N_j be the number of false rejections and R_j be the total number of rejections. We then have the following definitions:

Definition 14. We define the **estimated FDR** to be $\frac{1}{m}\sum_{j=1}^{m} \frac{N_j}{\max(R_j,1)}$ and we define the **estimated average power** to be $\frac{1}{mk}\sum_{j=1}^{m} \hat{k}_j - N_j$.

We generate all of our simulations independently of each other and in the same way so it follows that for each procedure our estimates of k will be i.i.d. The *FDR* is finite and is the expected value of the estimated *FDR*. So the estimated *FDR* will converge to *FDR* by the Strong Law of Large Numbers as $n \to \infty$. Similarly the estimated average power will tend to the expected value of the average power. Thus simulating p-values and calculating the estimated *FDR* and estimated Average Power for each procedure will allow us to compare performance. We do this in the 3 tables where we consider each of the difficulty settings in turn. From these tables it is clear that the proposed procedures are much more powerful than *Forward.Stop*, while still controlling the *FDR* to a given level.

Note that in the tables we have included an estimate for the standard error in the calculation of the Average Power. This is included in the Estimated SD of AP column. We have not done this for the Estimated *FDR* because the *FDR* is defined as an expectation.

To further illustrate how the procedures perform we have graphed histograms of the estimated value of k in Figures 15, 16 and 17 for a selection of the procedures. The true value of k is 20, so from these graphs we see that Normal.Stop and Changepoint.Stop are much better at estimating k than Forward.Stop or Strong.Stop. This is especially true in the Hard and Medium settings. In the Easy setting there is less of a difference.

It should be noted that the parameters below have been chosen to control the *FDR* using the results from Theorems 13 and 14. They have not been tailored to this particular situation. It would be interesting to see if different choices of parameters worked better in different situations but we have not considered this. As an example, to show that Normal.Stop(0.15, 3) control the *FDR* to a level 0.05 we can apply Theorem 14 to show that it controls the *FDR* to a level $\sum_{i=1}^{m-k} i a_{i+1,n}/(k+i)$ where $q = 0.15, n = 3, k = 20$ and $m = 100$. Using *R* we can show that this equals 0.0489. So Normal.Stop(0.15, 3) controls the *FDR* to this level which is less than 0.05.

Note that in the tables below all estimates have been rounded to 4 decimal places. Note also that a few of the estimated *FDR*s are greater than 0.05, this is just by random chance as convergence in the Strong Law of Large numbers occurs in the limit.

Algorithm	*FDR* controlled to level	Estimated *FDR*	Estimated Average Power	Estimated SD of AP
Uniform.Stop(0.05)	0.05	0.0000	0.0582	0.0258
Forward.Stop(0.05)	0.05	0.0000	0.0851	0.0864
Strong.Stop(0.05)	0.05	0.0000	0.1949	0.0659
Normal.Stop(0.15, 3)	0.05	0.0320	0.8418	0.2895
Normal.Stop(0.1, 4)	0.05	0.0270	0.7969	0.3086
Extended.Stop(0.27, 3, 3)	0.05	0.0483	0.9166	0.0928
Extended.Stop(0.18, 3, 4)	0.05	0.0436	0.8940	0.1231
Extended.Stop(0.1435, 4, 5)	0.05	0.0430	0.8552	0.1366
Extended.Stop(0.16, 5, 5)	0.05	0.0445	0.8371	0.1337
Decrease.Stop(2)	N/A	0.0236	0.6724	0.4037
Decrease.Stop(3)	N/A	0.0807	0.9224	0.2381
Decrease.Stop(4)	N/A	0.1476	0.9827	0.1100
Changepoint.Stop	N/A	0.0641	0.9714	0.0803

Table 1: Hard Setting: $B = 8$. The estimates are calculated after 2000 simulations.

Algorithm	*FDR* controlled to level	Estimated *FDR*	Estimated Average Power	Estimated SD of AP
Uniform.Stop(0.05)	0.05	0.0000	0.0784	0.0602
Forward.Stop(0.05)	0.05	0.0006	0.3003	0.3202
Strong.Stop(0.05)	0.05	0.0000	0.2805	0.0612
Normal.Stop(0.15, 3)	0.05	0.0447	0.9867	0.0761
Normal.Stop(0.10, 4)	0.05	0.0412	0.9647	0.1300
Extended.Stop(0.27, 3, 3)	0.05	0.0496	0.9272	0.0676
Extended.Stop(0.18, 3, 4)	0.05	0.0502	0.9167	0.0743
Extended.Stop(0.1435, 4, 5)	0.05	0.0449	0.8780	0.0940
Extended.Stop(0.16, 5, 5)	0.05	0.0475	0.8546	0.1129
Decrease.Stop(2)	N/A	0.0174	0.6779	0.4100
Decrease.Stop(3)	N/A	0.0546	0.9313	0.2307
Decrease.Stop(4)	N/A	0.1019	0.9872	0.0926
Changepoint.Stop	N/A	0.0321	0.9844	0.0458

Table 2: Medium Setting: $B = 14$. The estimates are calculated after 2000 simulations.

Algorithm	*FDR* controlled to level	Estimated *FDR*	Estimated Average Power	Estimated SD of AP
Uniform.Stop(0.05)	0.05	0.0000	0.1305	0.1286
Forward.Stop(0.05)	0.05	0.0097	0.8814	0.2737
Strong.Stop(0.05)	0.05	0.0000	0.3518	0.0564
Normal.Stop(0.15, 3)	0.05	0.0493	0.9988	0.0181
Normal.Stop(0.10, 4)	0.05	0.0478	0.9968	0.0134
Extended.Stop(0.27, 3, 3)	0.05	0.0482	0.9251	0.0680
Extended.Stop(0.18, 3, 4)	0.05	0.0505	0.9207	0.0699
Extended.Stop(0.1435, 4, 5)	0.05	0.0489	0.8857	0.0917
Extended.Stop(0.16, 5, 5)	0.05	0.0509	0.8624	0.1116
Decrease.Stop(2)	N/A	0.0113	0.7067	0.4121
Decrease.Stop(3)	N/A	0.0343	0.9360	0.2256
Decrease.Stop(4)	N/A	0.0641	0.9888	0.0881
Changepoint.Stop	N/A	0.0150	0.9913	0.0305

Table 3: Easy Setting: $B = 23$. The estimates are calculated after 2000 simulations.

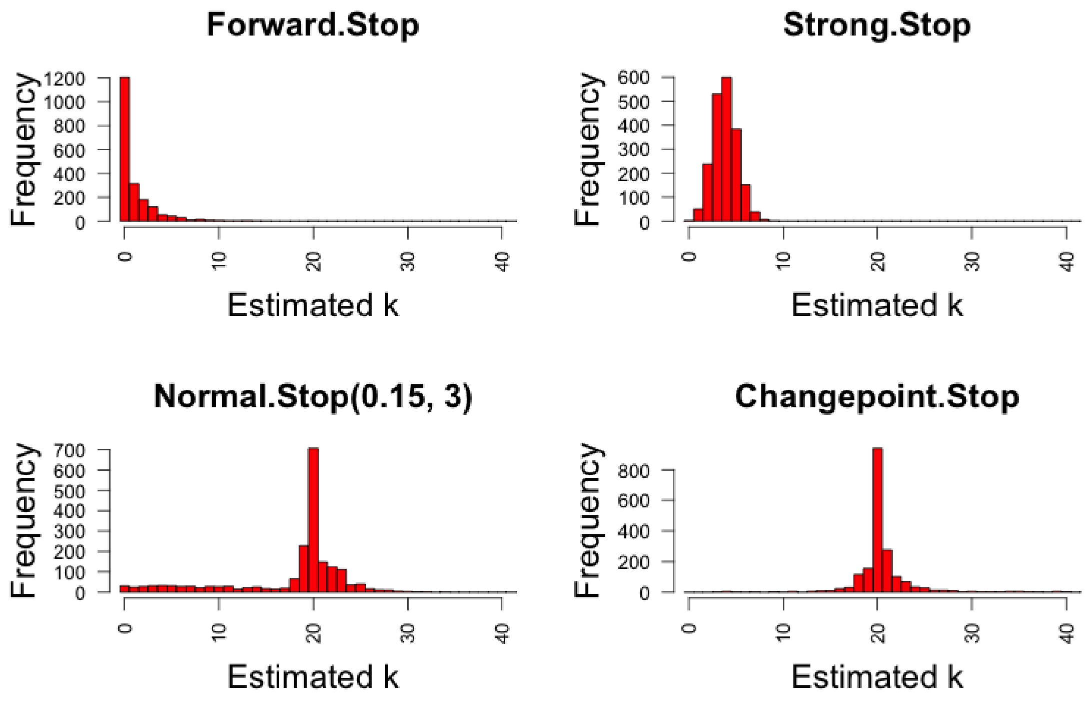


Figure 15: Hard Setting: $B = 8$. Histograms of the estimated values of k in the given procedures calculated after 2000 simulations.

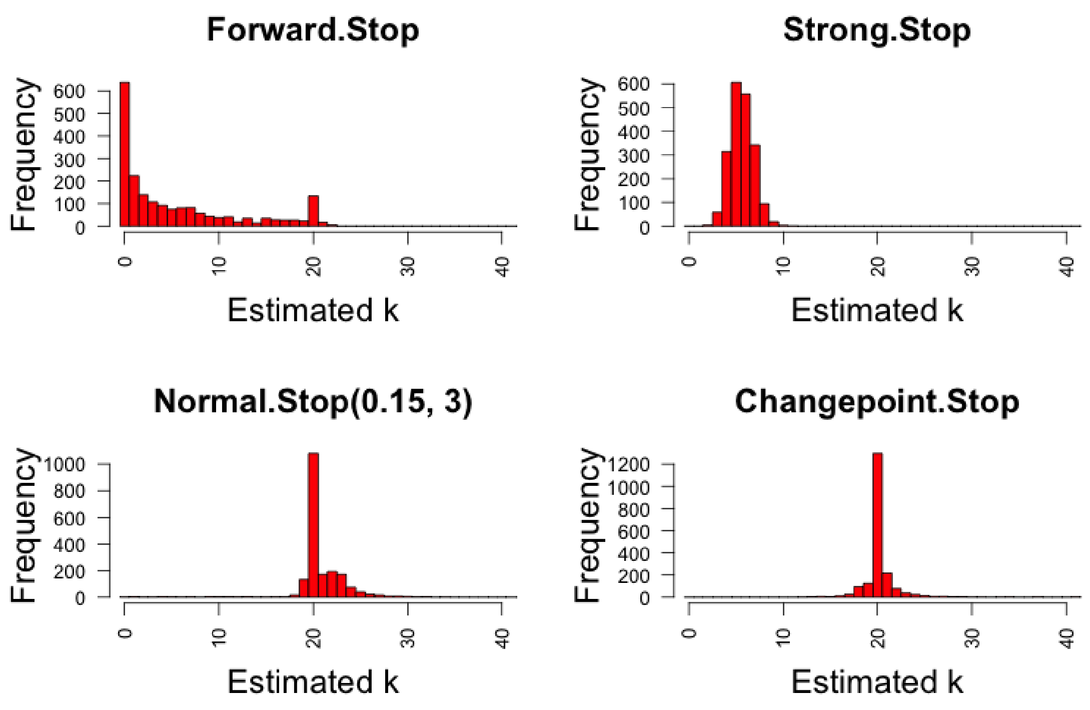


Figure 16: Medium Setting: $B = 14$. Histograms of the estimated values of k in the given procedures calculated after 2000 simulations.

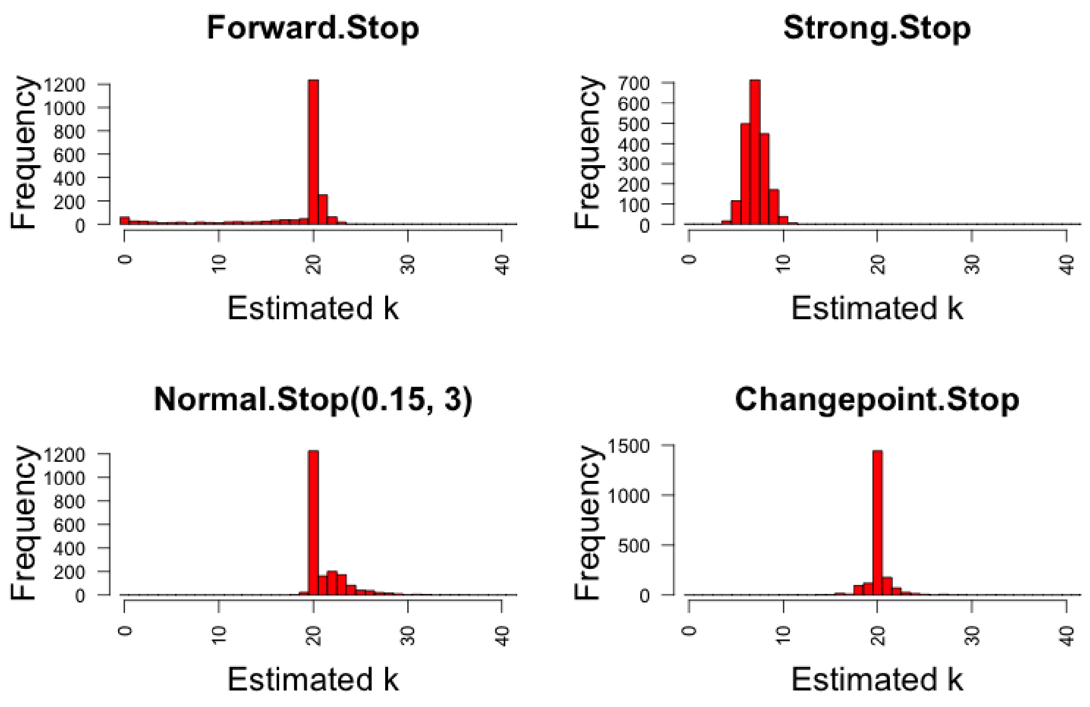


Figure 17: Easy Setting: $B = 23$. Histograms of the estimated values of k in the given procedures calculated after 2000 simulations.

5 Conclusion

We have discussed a number of techniques in this essay, many of which have applications to neuroscience. However, they are all interesting in their own right and have wide-ranging applications. There are a number of potential areas for future research.

In Chapter 2 we discussed general techniques for changepoint detection and the application of this to fMRI data via the NCPD algorithm. We also considered the approach of bootstrapping everything in order to avoid over-significance problems and misidentification of changepoints. There is a lot of scope for more work to be done in this context. One aspect that future research could look at would be to prove consistency results for applying the stationary bootstrap to the maximal CUSUM statistic and the minimal γ_k. This theory does not yet exist and it would be interesting (though possibly difficult) to explore. Another extension that would be interesting would be to consider replacing the stationary bootstrap with a permutation test; which would rearrange the data at random. Admittedly, when we apply the permutation test we usually require independence in the data; which we certainly do not have here. However, this may not be that much of an issue as the important thing is that, under the null hypothesis, resampling should result in data that looks similar to the original distribution. On the other hand when there is a changepoint, resampling should result in data that looks different than the observed data. The permutation test will do this and so is a viable alternative.

We have discussed methods such as *Forward.Stop* for dealing with Sequential Hypothesis Testing. These methods require independence assumptions on the p-values. The trouble is that in many important cases which fall into the category of Sequential Hypothesis Testing, the p-values will be dependent. It would be very useful to develop methods for controlling the *FDR* in this case, because of the applications to variable selection. Having developed such an algorithm for dealing with dependent p-values we would then be able to apply the mapping technique of Davison (2015) in order to have a method for controlling the *FDR* in dependent trees. The mapping technique hasn't been investigated much itself so that is also an area for potential research.

As it stands we have discussed a method *TreeShuffle* for dealing with dependent trees. This performs comparably to the Benjamini-Hochberg procedure under the assumption of positive-dependence (in fact we showed that it is slightly more powerful while still controlling the *FDR*). Further, the us of Benjamini-Yekutieli as the multiple testing procedure allows arbitrary dependence between the p-values. However, we think it is possible to do much better and to take advantage of the structure of the tree to create procedures which are more powerful while still controlling the *FDR*; even in the cases where there is dependence among the p-values.

We also considered the *False then True* problem. We described methods that controlled the *FDR* the assumption that: p_i, $i \in I_0$ are independent, and independent of $\{p_i : i \notin I_0\}$. Note also that the *False then True* problem is a subset of the *False then True* hypothesis tree problem so methods for controlling the *FDR* for dependent trees extend to *False then True*, though would not necessarily be that powerful.

We also discussed *Changepoint.Stop* as a method. This method performs very well, and it would be useful to derive some results concerning how it controls the *FDR* and similarly with regard to *Decrease.Stop*. We also looked at the more general problem of identifying changepoints in distribution and introduced *KSS* as a means for doing so. There is a lot of scope for further research here, looking at new methods and proving consistency results about existing methods. Some other things to consider are analysis of multiple changepoints in distribution and considering means of identifying whether a given changepoint is significant (using bootstrapping techniques for instance). More generally, it would be interesting to extend the methods for analysing a change in distribution from the 1-dimensional setting to the multi-dimensional setting.

All of the methods for controlling the *FDR* in trees have potential applications to analysing structural breaks in fMRI data. This is because Binary Segmentation with hypothesis testing can be used to identify the structural breaks. Algorithms such as NCPD rely on this and so discussing hypothesis testing for trees is very relevant in this context. Applications of changepoint detection extend far beyond neuroscience and so the techniques discussed here have the potential to be applied much more widely.

References

[1] J. Aston and C. Kirch. Change points in high dimensional settings. arXiv preprint arXiv:1409.1771, 2014.

[2] Anderson, T. W. On the Distribution of the TwoSample CramerVon Mises Criterion. Annals of Mathematical Statistics 33, pp. 11481159. (1962)

[3] Y. Benjamini and Y. Hochberg, Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. Journal of the Royal Statistical Society. Series B (methodological), 57(1), 289300, 1995.

[4] Benjamini, Yoav; Yekutieli, Daniel. The control of the false discovery rate in multiple testing under dependency. Ann. Statist. 29, no. 4, 1165–1188. doi:10.1214/aos/1013699998, 2001.

[5] Bonferroni, C. E. "Teoria statistica delle classi e calcolo delle probabilit." Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze 8, 3-62, 1936.

[6] K. Chen and J. Lei. Network cross-validation for determining the number of communities in network data. axXiv preprint arXiv: 1411.1715, 2014.

[7] I. Cribeen and Y. Yu Estimating whole brain dynamics using spectral clustering. arXiv preprint arXiv: 1509.03730, 2016.

[8] Davison, A. An unpublished report: Change point analysis in fMRI Data (2015).

[9] Efron, B. Bootstrap methods: another look at the jackknife. Ann. Statist. 7, 126, 1979.

[10] Liu, Fang, and Sanat K. Sarkar. "A new adaptive method to control the false discovery rate." Recent Advances in Biostatistics: False Discovery Rates, Survival Analysis, and Related Topics. Series in Biostatistics 4 (2011): 3-26.

[11] D. Franco Saldana, Y. Yu, and Y. Feng. How many communities are there? axXiv preprint arXiv: 1412.1684, 2014.

[12] Fryzlewicz, P. Wild Binary Segmentation for multiple change-point detection. Ann. Statist., 42(6), pp.2243-2281, 2014.

[13] G'Sell, Max Grazier, et al. Sequential selection procedures and false discovery rate control. Journal of the Royal Statistical Society: Series B (Statistical Methodology) (2015).

[14] Heiserman, Joseph E. "Measurement error of percent diameter carotid stenosis determined by conventional angiography: implications for noninvasive evaluation." American journal of neuroradiology 26.8 (2005): 2102-2107.

[15] Holm, Sture. "A simple sequentially rejective multiple test procedure." Scandinavian journal of statistics (1979): 65-70.

[16] Kifer, Daniel, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. Proceedings of the Thirtieth international conference on Very large data bases-Volume 30. VLDB Endowment, 2004.

[17] Kunsch, H.R. The jackknife and the bootstrap for general stationary observations. Ann. Statist. 17 12171241, 1989.

[18] M. Lavielle and E. Moulines. Least-squares estimation of an unknown number of shifts in a time series. J. Time Ser. Anal., 21:3359, 2000.

[19] Lindquist, Martin A. "The statistical analysis of fMRI data." Statistical Science 23.4 (2008): 439-464.

[20] Marcus, R., P. Eric, and K. R. Gabriel. On closed testing procedures with special reference to ordered analysis of variance. Biometrika 63 (3), 655660. 1976.

[21] Newman, M. E. J. (2004) Detecting community structure in networks. EPJ B, 38(2), 321-330.

[22] Seiji Ogawa, Tso-Ming Lee, Alan R Kay, and David W Tank. Brain magnetic resonance imaging with contrast dependent on blood oxygenation. Proceedings of the National Academy of Sciences, 87(24):98689872, 1990.

[23] Politis, D. N., and Romano, J. P. The Stationary Bootstrap. Journal of the American Statistical Association, 89(428), 13031313. http://doi.org/10.2307/2290993, 1994.

[24] Romano, Joseph P., Azeem M. Shaikh, and Michael Wolf. "Control of the false discovery rate under dependence using the bootstrap and subsampling." Test 17.3 (2008): 417-442.

[25] Ross, G. J, and Adams N.M. "Two nonparametric control charts for detecting arbitrary distribution changes." Journal of Quality Technology 44.2 (2012): 102.

[26] Rajen. S. Modern Statistical Methods Notes. www.statslab.cam.ac.uk/∼ rds37/modern_stat_methods.html (2015).

[27] Sarkar, S. K. (2002). Some results on false discovery rate in stepwise multiple testing procedures. Annals of Statistics 30, 239257.

[28] Sarkar, S. K. (2008b). On methods controlling the False Discovery Rate (with discussions). Sankhya 70, 135168.

[29] E. S. Venkatraman. Consistency results in multiple change-point problems. Technical Report No. 24, Department of Statistics, Stanford University, available from https://statistics.stanford.edu/ resources/technical-reports, 1992.

[30] Von Luxburg, Ulrike. "A tutorial on spectral clustering." Statistics and computing 17.4 (2007): 395-416.

[31] , J.A. (1996) Weak Convergence and Empirical Processes, Springer. ISBN 0-387-94640-3.

[32] L. Vostrikova. Detecting disorder in multidimensional random processes. Soviet Math. Dokl., 24:5559, 1981.

[33] Y.-C. Yao and S. T. Au. Least-squares estimation of a step function. Sankhya Series A, 51:370381, 1989.

Appendix A - Proof of Theorem 1

Lemma 4. *Let Y be a random variable with continuous cdf: F and let Z be another random variable, independent of Y. Then the cdf of $Y + Z$ is continuous and given any constant $c \in \mathbb{R}, cY$ has a continuous cdf.*

Proof: Given $t \in \mathbb{R}$, let $H(t) := \mathbb{P}(Y+Z \leq t) = \mathbb{E}\,(\,1\,[Y + Z \leq t]\,) = \mathbb{E}\,(\,\mathbb{E}\,[\,1\,[Y + Z \leq t] \mid Z\,]\,) = \mathbb{E}\,(\,F(t - Z)\,)$. (As Y and Z are independent). Now, given a sequence, $t_n \to t$ let $f_n(z) = F(t_n - z)$ and $f(z) = F(t - z)$. Then for all $z, f_n(z) \to f(z)$ as $n \to \infty$ (as F is continuous). Also $|f_n| \leq 1$ which is integrable and the f_n are measurable as they are continuous. So the dominated convergence theorem implies that $\mathbb{E}\,(\,f_n(Z)\,) \to \mathbb{E}\,(\,f(Z)\,)$ as $n \to \infty$. So for all sequences $t_n \to t, H(t_n) \to H(t)$ as $n \to \infty$. So it follows that H is continuous.
The last aspect of the lemma is obvious. □

Lemma 5. *If Y has continuous CDF: F then given $t \in \mathbb{R}$, $\mathbb{P}(Y \leq t) = \mathbb{P}(Y < t)$ and $\mathbb{P}(Y = t) = 0$.*

Proof: Given $t \in \mathbb{R}$, for all $\epsilon > 0$, $\mathbb{P}(Y \leq t - \epsilon) \leq \mathbb{P}(Y < t) \leq \mathbb{P}(Y \leq t)$.
Thus: $F(t - \epsilon) \leq \mathbb{P}(Y < t) \leq F(t)$. So taking $\epsilon \to 0$, continuity implies that $\mathbb{P}(Y < t) = F(t) = \mathbb{P}(Y \leq t)$ for all t. From this it clearly follows that $\mathbb{P}(Y = t) = 0$. □

Lemma 6. *Let Y be a random variable with continuous CDF: F, and let G be the CDF of $|Y|$, then F continuous implies that G is continuous.*

Proof: Given $t \in \mathbb{R}$, $G(t) = \mathbb{P}(|Y| \leq t) = \mathbb{P}(-t \leq Y \leq t) = \mathbb{P}(Y \leq t) - \mathbb{P}(Y < -t)$. This equals: $\mathbb{P}(Y \leq t) - \mathbb{P}(Y \leq -t) = F(t) - F(-t)$ by applying Lemma 5. This is the difference of two continuous functions and so is continuous. □

Lemma 7. *For $i = 1, \ldots, n$ (some $n \in \mathbb{N}$), let $Y_i \sim F_i$ be real random variables (which are not necessarily independent) for some CDFS: F_i which are continuous. Let $Y_{max} = max_{i \in \{1,\ldots,n\}}\{Y_i\}$, and let Y_{max} have some CDF: F. Then F is continuous.*

Proof: F is right-continuous as it is a CDF, so it suffices to establish left-continuity. For this, suppose we have $\epsilon > 0$, then given $x \in \mathbb{R}, |F(x) - F(x - \epsilon)| = F(x) - F(x - \epsilon) = \mathbb{P}(x - \epsilon < Y_{max} \leq x) \leq \sum_{i=1}^{n} F_i(x) - F_i(x - \epsilon) = \sum_{i=1}^{n} |F_i(x) - F_i(x - \epsilon)| \to 0$ as $\epsilon \to 0$ by continuity of the F_i. □

Appendix B - Midpoint Correction

One further improvement to changepoint procedures in general that we would like to briefly mention is that of the midpoint correction. Let the estimates of the changepoints be : $\eta_1, ..., \eta_{\hat{N}}$, where $\hat{N}$ is the estimated number of changepoints and set $\eta_0 = 1$ and $\eta_{\hat{N}+1} = T$. Then an improved set of estimates can be derived by re-estimating the location of each of the changepoints: η_i by maximizing CUSUM (or minimal γ) statistic on $[a_{i-1}, a_i]$ where a_j is the midpoint of the interval $[\hat{\eta}_j, \hat{\eta}_{j+1}]$. This will likely reduce bias. We have not explored this particular concept in greater detail. However future research could look at how it can be exploited.